

LDMAT: Função de Custo Baseada em Matrizes de Distâncias para o Treinamento de Redes Neurais Convolucionais

Eduardo da Silva Ribeiro

Programa de Pós-Graduação em Engenharia Elétrica

Universidade Federal de Minas Gerais

Orientador: Prof. Dr. Antônio de Pádua Braga

Tese

Doutorado em Engenharia Elétrica

17/12/2021

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

**LDMAT: FUNÇÃO DE CUSTO BASEADA EM MATRIZES DE
DISTÂNCIAS PARA O TREINAMENTO DE REDES NEURAIS
CONVOLUCIONAIS**

Eduardo da Silva Ribeiro

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Doutor em Engenharia Elétrica.

Orientador: Dr. Antônio de Pádua Braga

Belo Horizonte – MG

Dezembro de 2021

R484I

Ribeiro, Eduardo da Silva.

LDMAT [recurso eletrônico]: função de custo baseada em matrizes de distâncias para o treinamento de redes neurais convolucionais / Eduardo da Silva Ribeiro. - 2021.

1 recurso online (83 f. : il., color.) : pdf.

Orientador: Antônio de Pádua Braga.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f. 76-83.

Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Redes Neurais (Computação) – Teses. 3. Matrizes (Matemática) – Teses. I. Braga, Antônio de Pádua. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

"Ldmat: Função de Custo Baseada Em Matrizes de Distâncias Para O Treinamento de Redes Neurais Convolucionais"

Eduardo da Silva Ribeiro

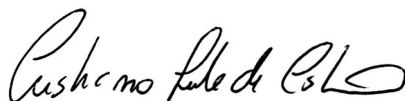
Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica.

Aprovada em 17 de dezembro de 2021.

Por:



Prof. Dr. Antônio de Pádua Braga
DELT (UFMG) - Orientador



Prof. Dr. Cristiano Leite de Castro
(UFMG)



Prof. Dr. Frederico Gadelha Guimarães
DEE (UFMG)



Prof. Dr. Alexei Manso Correa Machado
Departamento de Ciência da Computação (PUC-MG)



Prof. Dr. Raul Fonseca Neto
Departamento de Ciência da Computação (UFJF)

Resumo

As Redes Neurais Convolucionais (CNNs) têm estado na vanguarda em pesquisas de redes neurais nos últimos anos, seu desempenho inovador em áreas como classificação de imagens reuniu esforços para o desenvolvimento de novas arquiteturas, porém recentemente surgiu uma busca por novas funções custo para CNNs. A função de custo de entropia cruzada (*softmax loss*) vem sendo amplamente utilizada devido a sua eficiência na separação de classes, entretanto existe uma deficiência em projetar uma margem adequada de separação das classes e compacidade intraclasse das características. Embora alguns estudos tenham abordado esse problema, a maioria das soluções apresentam refinamentos da função de custo de entropia cruzada ou realizam a combinação da mesma com outros componentes para melhorar a generalização do modelo. Nesta tese apresenta-se uma função de custo nova baseada em matrizes de distâncias (LDMAT) que faz análise pareada de um conjunto de características e opera diretamente nas características extraídas pelos filtros convolucionais, permitindo assim, seu uso em classificadores arbitrários. É apresentado também um método de regularização pela inserção de uma perturbação na matriz de distâncias de rótulos, tendo desempenho similar ao desligamento de conexões (*dropout*). A abordagem proposta permite a combinação de modelos treinados com a função de custo LDMAT com diversos níveis de regularização, o que melhora a generalização do modelo final. Para validar a função de custo proposta, experimentos foram realizados para demonstrar a sua eficiência no treinamento de CNNs em tarefas de classificação. Nos conjuntos de dados FMNIST, EMNIST, CIFAR₁₀, CIFAR₁₀₀ e SVHN a função de custo LDMAT supera outras abordagens que utilizam arquiteturas similares, e na base MNIST os resultados são comparáveis com outras funções de custo propostas na literatura.

Palavras-chaves: Função de Custo, Redes Neurais Convolucionais, Matrizes de distâncias.

Abstract

Convolutional Neural Networks (CNNs) have been on the forefront of neural network research in recent years. Their breakthrough performance in fields such as image classification has gathered efforts in the development of new CNN-based architectures, but recently a search for new loss functions for CNNs has emerged. Softmax loss remains the most popular loss function due mainly to its efficiency in class separation, but the function is unsatisfactory in terms of intra-class compactness. In this thesis, a new loss function based on distance matrices (LDMAT) is presented, which performs a pairwise analysis of a set of features and operates directly on the features extracted by convolutional filters, thus allowing its use in arbitrary classifiers. A regularization method by inserting a disturbance in the label distance matrix is also presented, with performance similar to dropout. The proposed approach allows the combination of trained models with the LDMAT loss function with different levels of regularization, which improves the generalization of the final model. In order to validate the proposed loss function, experiments were performed to demonstrate its efficiency in training CNNs during classification tasks. In the FMNIST, EMNIST, CIFAR₁₀, CIFAR₁₀₀ and SVHN datasets, the LDMAT loss function outperforms other approaches that use similar architectures, and in the MNIST dataset, the accuracy were close to results of other loss functions proposed in the literature.

Keywords: Loss Function, Convolutional Neural Network, Distances Matrices.

Lista de Figuras

2.1	Arquitetura de rede LeNet-5. Fonte: Lecun et al. (1998)	21
2.2	Funções de ativação: Sigmoid e ReLU. Fonte: Activation Functions in Neural Networks	24
2.3	Regularização com desligamento aleatório de conexões (<i>dropout</i>), fonte: Srivastava et al. (2014)	24
2.4	ZefNet. Fonte: Zeiler & Fergus 2014	25
2.5	Arquitetura da rede VGG-16. Fonte: VGG in TensorFlow	25
2.6	Subamostragem de média global. Fonte: Global Average Pooling	26
2.7	Bloco <i>Inception</i> . Fonte: Szegedy et al. (2015)	26
2.8	Variações do bloco <i>Inception</i> . Fonte: Szegedy et al. (2016)	27
2.9	Bloco Residual. Fonte: He et al. (2016)	29
2.10	ResNext. Fonte: Xie et al. (2017)	30
2.11	DenseNets. Fonte: Huang et al. (2017)	30
2.12	Bloco <i>Xception</i> . Fonte: Chollet (2016)	32
2.13	Bloco-SE. Fonte: Hu et al. (2018)	33
3.1	Superfície de separação da <i>softmax loss</i> na base de dados MNIST. Para gerar a figura foi utilizada a base de treinamento em uma arquitetura de CNN com camada totalmente conectada com dois neurônios antes da camada de saída. Desta forma é possível ter uma visualização 2D do espaço de características. Fonte: Qin et al. (2020)	36
3.2	Efeito das funções de custo na distribuição das características no espaço de separação. Em (a) <i>Softmax loss</i> , (b) <i>L2-softmax loss</i> (Ranjan et al., 2017). (c) <i>L-Softmax</i> (Liu et al., 2016) e (d) <i>AM-Softmax</i> (Wang et al., 2018a). Figura adaptada de Liang et al. (2020)	39
3.3	Espaço de separação GCPL gerado no conjunto de dados MNIST, foi utilizado $\lambda = 0.001$. Observe que existe uma melhor compacidade intraclasse, porém o espaço de separação ainda é angular.	40
3.4	Diferença do funcionamento das redes SCNet e PGL, fonte: (Qin et al., 2020)	43

3.5	Esquema de funcionamento da função de custo com sinais locais de erro, fonte: (Nøkland & Eidnes, 2019)	44
4.1	Exemplo de matrizes de distâncias, a) obtida pelas características e b) obtidas pelos rótulos.	49
4.2	Exemplo de representação de características de 3 classes em um espaço bidimensional.	50
4.3	Matrizes de distâncias obtidas pelo exemplo da Figura 4.2, onde a) é matriz de distância das características, b) a matriz de distâncias dos rótulos e c) a matriz de distâncias dos rótulos com um nível de perturbação.	50
4.4	Visualização das características obtidas em dois treinamentos distintos: a) obtida a <i>Softmax Loss</i> e b) obtida com o treinamento da LDMAT. A dimensionalidade das características foi reduzida utilizando a redução de dimensionalidade t-SNE (Belkina <i>et al.</i> , 2019). Foi utilizada o conjunto de dados CIFAR10 e o treinamento foi feito com a arquitetura LiuNet-B.	52
4.5	Diagrama ilustrando a diferença entre o modo de treinamento de uma CNN padrão e um treinamento em duas etapas utilizando a função de custo LDMAT. Os blocos em rosa indicam as camadas convolucionais, o retângulo em azul indica as características extraídas pelos blocos convolucionais, o retângulo em verde indica as camadas totalmente conectadas e o quadrado em vermelho indica a função de custo <i>softmax loss</i> . A segunda figura indica que o treinamento utilizando a LDMAT é feito em duas etapas distintas indicado pela separação pelas linhas tracejadas. As duas matrizes em laranja e amarelo indicam as matrizes de distâncias de características e de rótulos, utilizadas pelo função de custo LDMAT.	53
5.1	Imagens para ilustrar os conjuntos de dados utilizados nos experimentos: MNIST, <i>Fashion MNIST</i> , <i>Extended MNIST</i> , <i>Street View House Number</i> , CIFAR10 e CIFAR100	59
5.2	Curvas de convergência da função de custo LDMAT com variações do nível de perturbação na matriz de distância de rótulos e o uso do <i>dropout</i> . Foi utilizado o conjunto de dados CIFAR10 com aumento de dados e a rede foi treinada com a arquitetura LiuNet-B.	65

5.3	Acurácia obtida ao longo do treinamento de uma CNN com a função de custo LDMAT. Foi utilizada o conjunto de dados CIFAR10 com aumento de dados e a arquitetura LiuNet-B. Os pesos foram salvos a cada 100 épocas e posteriormente as características foram obtidas e utilizadas em um classificador (<i>perceptron</i> linear) para a obtenção da acurária.	66
5.4	Histograma dos rótulos do conjunto de imagens SVHN, referente a partição de treinamento e sem uso do conjunto de dados extras.	71

Lista de Tabelas

5.1	Descrição dos conjuntos de imagens utilizados no experimentos.	58
5.2	Arquitetura utilizada nos experimentos. Proposta no trabalho de Liu <i>et al.</i> (2016), chamou-se esta arquitetura de LiuNet e esta foi utilizada como critério de comparação por diversos trabalhos publicados. Por exemplo $[3 \times 3, 64]_3$ denota 3 camadas convolucionais com 64 filtros de tamanho 3×3	60
5.3	Arquitetura utilizada nos experimentos, Proposta no trabalho de Liu <i>et al.</i> (2016) e utilizada com conjunto de dados CIFAR100.	61
5.4	Análise do número de parâmetros da rede LiuNet em relação ao número de operações de ponto flutuante por segundo (FLOPS) na escala de 10^9 e o tempo gasto para a execução de uma época treinamento (executado em uma GPU Tesla K80). Para treinar a rede LiuNet-A usou-se o conjunto de dados MNIST e LiuNet-B usou-se CIFAR10. A indicação s/FC significa que a arquitetura de rede utilizada não tem a camada totalmente conectada e a camada de saída.	62
5.5	Escolha da combinação das técnicas de desligamento de conexões e perturbação na matriz de distâncias de rótulos. Foi utilizada o conjunto de dados CIFAR10+ e rede LiuNet-B. Drop ₁ , Drop ₂ e Drop ₃ representam o nível de desligamento de conexões no final das camadas convolucionais como exibido nas Tabelas 5.3 e 5.3. ξ representa o nível de perturbação na matriz de distâncias de rótulos da LDMAT. A coluna acc indica a acurácia obtida pela combinação de parâmetros de regularização na etapa de classificação.	64
5.6	Relação das características utilizadas para o conjunto de soluções. Nas primeiras linhas são apresentadas a taxa de acurácia individual de cada combinação de parâmetros e na última linha a acurácia com a fusão das características.	66
5.7	Resultados publicados que utilizam o conjunto de dados MNIST e os resultados da função de custo LDMAT.	67
5.8	Resultados com conjunto de dados <i>Extended MNIST</i>	68
5.9	Resultados com conjunto de dados <i>Fashion MNIST</i>	68

5.10	Resultados publicados com o conjunto de dados CIFAR10 e os resultados obtidos com a função de custo LDMAT. A indicação + sinaliza o uso de aumento de dados na base de dados.	69
5.11	CIFAR100	70
5.12	Resultados com conjunto de dados SVHN. A coluna SVHN-ext, indica que o resultado foi obtido com a utilização do conjunto de dados extras para o treinamento.	71

Lista de Abreviaturas

CNN	<i>Convolutional Neural Networks</i>
ILSVRC	<i>ImageNet Large Scale Visual Recognition Challenge</i>
SVM	<i>Support Vector Machines</i>
LDA	<i>Discriminant Linear Analysis</i>
2D	<i>Two Dimensional</i>
LDMAT	<i>Loss Function Based on Distance Matrices</i>
RGB	<i>Red, Green, Blue</i>
ReLU	<i>Rectified Linear Units</i>
VGG	<i>Visual Geometry Group</i>
CDF	<i>Cumulative Distribution Function</i>
GCPL	<i>Generalized Convolutional Prototype Learning</i>
PEDCC	<i>Predefined Evenly-Distributed Class Centroids</i>
CCL	<i>Constrained Center Loss</i>
SGD	<i>Stochastic gradient descent</i>
PGL	<i>Pairwise Gaussian Loss</i>
SCNet	<i>Separability and Compactness Network</i>
MSE	<i>Mean Squared Error</i>
RMSE	<i>Root Mean Squared Error</i>
FLOPS	<i>Floating Point Operations Per Second</i>
MNIST	<i>Modified National Institute of Standards and Technology</i>
SVHN	<i>Street View House Number</i>
EMNIST	<i>Extended MNIST</i>
FMNIST	<i>Fashion MNIST</i>
GPU	<i>Graphics Processing Unit</i>
FC	<i>Full Connected Layer</i>
t-SNE	<i>t-Distributed Stochastic Neighbor Embedding</i>

Sumário

Lista de Figuras	viii
Lista de Tabelas	x
Lista de Abreviaturas	xi
1 Introdução	14
1.1 Hipótese	17
1.2 Proposta	17
1.3 Objetivos	18
1.4 Contribuições	19
2 Arquiteturas de Redes Neurais Convolucionais	20
2.1 Histórico das Redes Neurais Convolucionais	20
2.2 Funcionamento da Camada Convolucional	21
2.3 Arquiteturas de redes neurais convolucionais	23
2.3.1 Abordagens de exploração espacial	24
2.3.2 Exploração da profundidade	28
2.3.3 Abordagens com múltiplas conexões baseadas na largura	31
2.3.4 Exploração do mapa de características	33
2.3.5 Conclusões do capítulo	34
3 Funções de Custo para CNNs	35
3.1 Funções de Custo Independentes da <i>Softmax Loss</i>	36
3.2 Refinamentos da <i>Softmax Loss</i>	38
3.3 <i>Softmax loss</i> combinada com outras funções	39
3.4 Resumo das funções de custo	46
4 LDMAT	48
4.1 Otimização	54
4.2 Algoritmo	54
4.3 Estratégia de combinação (<i>ensemble</i>)	55
4.4 Conclusão do Capítulo	56

5	Experimentos	57
5.1	Conjuntos de Dados	57
5.2	Arquiteturas	60
5.3	Detalhes de Implementação	62
5.4	Acurácia	67
5.5	Discussão	71
6	Conclusões	74
6.1	Trabalhos Futuros	75

Capítulo 1

Introdução

Na década de 2010 houve um expressivo crescimento no uso das redes neurais profundas (*deep neural networks*), cujo bom desempenho possibilitou o avanço do estado-da-arte em áreas como, por exemplo, classificação de imagens (Simonyan & Zisserman, 2015) (He *et al.*, 2016), processamento de linguagem natural (Jin *et al.*, 2020) e geração de imagens sintéticas (Jabbar *et al.*, 2021). Apesar de não ser uma técnica recente, o episódio que chamou atenção da comunidade científica para o potencial das redes neurais profundas foi a competição ILSVRC¹ de 2012, que foi vencida por Krizhevsky *et al.* (2012) que utilizou a rede neural convolucional (*Convolutional Neural Network* - CNN) AlexNet. Na competição ILSVRC equipes competiram apresentando propostas para a tarefa de classificar um volume grande de imagens pertencentes a mil classes. O uso de uma CNN era inédito na competição e os resultados obtidos foram consideravelmente superiores comparado com as abordagens utilizadas anteriormente. Ficou evidente a capacidade das CNNs de aprender boas características de imagens através de um modelo hierárquico com operações de convolução.

Apesar de Cybenko (1989) ter demonstrado que é possível aproximar qualquer função com uma rede neural com uma camada escondida, o uso dos *pixels* de uma imagem, sem nenhum pré-processamento, como entrada em um rede neural convencional não é uma abordagem eficiente porque um modelo de rede totalmente conectada trata todos os *pixels* de entrada de forma igual, ignorando a relação espacial entre eles. Desta forma, cria-se uma conectividade alta na rede, o que torna o espaço para o ajuste de parâmetros também com dimensão alta. Desta forma, torna-se difícil realizar o ajuste dos parâmetros durante o treinamento e chega-se a soluções com sobreajuste (*overfitting*). Antes das CNNs, a tarefa de classificação de imagens, na maioria das vezes, era feita em duas etapas: extração de características e classificação. Algoritmos específicos eram utilizados para a extração de características em cada problema, sendo que após a extração de características, a classificação era feita em uma

¹ImageNet Large Scale Visual Recognition Challenge - www.image-net.org

etapa separada com um método de classificação, entretanto, o desempenho do classificador era dependente do sucesso da etapa de extração de características. Para exemplificar, o desafio ILSVRC de 2011 foi vencido por uma abordagem em duas etapas, primeiramente as características das imagens foram extraídas a partir de um pré-processamento não-supervisionado com transformações e técnicas de compressão de dados, após isto o classificador de Máquina de Vetores de Suporte (*Support Vector Machines* - SVM) linear foi utilizado para discriminar as classes (Russakovsky *et al.*, 2014). O treinamento em duas etapas é importante porque tem-se um método específico para extração de características e outro para classificação. A dificuldade antes das CNNs era encontrar um método eficiente para a extração de características.

Antes das CNNs, o projeto da etapa de extração de características era empírico e seguia a percepção do projetista. Muitas vezes, eram considerados modelos que realizavam operações matemáticas que nem sempre favoreciam as características que possibilitavam boa discriminação de classes e compactação intra-classe. Uma abordagem comum utilizada para a extração de características em duas etapas é mapear os dados de entrada de dimensão alta em um espaço de dimensão mais baixa, como é feito na análise discriminante linear (*Discriminant Linear Analysis* - LDA) (Chang *et al.*, 2016), que reduz a dimensionalidade considerando a separação entre as classes. No LDA clássico utilizado em imagens é feita uma conversão para a forma vetorial, assim algumas informações espaciais da imagem podem ser perdidas. Existem entretanto abordagens do LDA que tratam diretamente imagens em forma de matriz, como o método de projeção *rank-k* composto (Chang *et al.*, 2016) e o método de análise discriminante não linear 2D convolucional (Wang *et al.*, 2021). Outras abordagens foram propostas especificamente para imagens como a transformação de características invariável a escala (*Scale Invariant Feature Transform* - SIFT) (Lowe, 2004) e a técnica do histograma orientado a gradiente (*Histogram of Oriented Gradients* - HOG) (Dalal & Triggs, 2005).

As redes neurais convolucionais utilizam o compartilhamento de pesos por meio dos filtros de convolução e realizam a extração de características de forma integrada com o classificador, na proposta original das arquiteturas, realizando somente uma etapa de treinamento. Os pesos das máscaras de convolução e das camadas totalmente conectadas são ajustados de forma supervisionada a partir de um método de treinamento guiado por uma função de custo utilizada globalmente no modelo (Lecun *et al.*, 1998).

Durante a fase de ascensão do uso das CNNs, uma sequência de melhorias foi proposta nas arquiteturas das redes neurais convolucionais. Em um primeiro momento buscou-se aumentar o desempenho por meio do aumento da profundidade das redes convolucionais, isto acabou gerando arquiteturas com muitos parâmetros

para serem ajustados, o que agravou o custo computacional para treinar os modelos propostos. Para contornar a necessidade de grande quantidade de imagens de treinamento foram propostas técnicas de aumento de dados como *random cropping* (Shorten & Khoshgoftaar, 2019) e *random erasing* (Zhong et al., 2020). Para evitar o sobreajuste do modelo foram propostas técnicas de regularização das arquiteturas como, por exemplo, desligamento de neurônios (Srivastava et al., 2014), normalização de lotes (Ioffe & Szegedy, 2015) e suavização de rótulos (Szegedy et al., 2016). O aumento da profundidade nos modelos criou o problema do desaparecimento do gradiente durante o treinamento, o que levou em segundo momento, as propostas de arquiteturas se concentrarem em tornar mais eficiente as conexões e criação de blocos, como os residuais (He et al., 2016).

Além dos diferentes tipos de conexões entre os blocos convolucionais e técnicas de regularização, explorados nas arquiteturas das CNNs, outro aspecto importante vem sendo investigado nos últimos anos: as funções de custo. A função de custo guia o ajuste dos pesos de todo o modelo e tem um papel importante porque define o espaço de projeção das características extraídas de acordo com os rótulos das suas respectivas classes. Na maioria das vezes os pesos de um modelo de redes neurais são ajustados por um processo de otimização, onde a técnica mais utilizada é o gradiente descendente.

A função de custo de entropia cruzada (divergência de Kullback-Leibler) (Kullback & Leibler, 1951) é a mais utilizada em propostas de modelos de redes neurais convolucionais. Como a função de custo de entropia cruzada é sempre utilizada em combinação com a função de ativação *softmax* também é conhecida como *softmax loss* e que tem como característica discriminar os atributos projetando-os em um espaço de coordenadas polares (Peng & Yu, 2021).

Apesar da rápida convergência, a *softmax loss* não realiza uma boa compacidade intraclasse das características extraídas, ocasionando dificuldades de generalização (Peng & Yu, 2021). A separação interclasse é desejável para que no espaço de classificação exista uma separação das classes. A compacidade ou compactação intraclasse busca que as características da mesma classe estejam agrupadas em uma região do espaço, o que é desejável para o modelo ter uma boa generalização para novas amostras. Classes com um espalhamento grande das características no espaço de separação podem ter dificuldade de generalização.

No trabalho de Liu et al. (2016) a *softmax loss* é reescrita em termos de similaridade do cosseno com amplitude e ângulo, onde o espaço de métrica é representado em coordenadas polares (Liu et al., 2016) (Peng & Yu, 2021).

Alguns pesquisadores trataram a dificuldade da *softmax loss* de gerar uma boa separação interclasse e compacidade intraclasse inserindo restrições na formulação da função de custo, como apresentado nos trabalhos de Liu et al. (2016), Wang et al.

(2018a) e Ranjan *et al.* (2017). Outras abordagens, utilizaram a *softmax loss* junto com outras funções de custo para projetar um espaço de características com melhor separação e compacidade intraclasse. Para realizar esta combinação algumas propostas utilizaram múltiplas funções de custo como feito por Xu *et al.* (2016), cada uma com uma penalidades diferentes, ou compõe a *softmax loss* com uma outra função de custo. Esta combinação na função de custo funciona como um termo de regularização, que algumas vezes, realiza o cálculo de distâncias pareadas entre amostras, como na PGL (Qin *et al.*, 2020) e SCNet (Zhou *et al.*, 2019). Estas abordagem de contraste entre amostras, ou distâncias de pares, se inspiram em proposta de funções de custo totalmente independentes da *softmax loss* como a *contrastive loss* (Hadsell *et al.*, 2006) e a *triplet loss* (Schroff *et al.*, 2015).

1.1 Hipótese

Nesta tese de doutorado, partiu-se da hipótese que distâncias entre amostras podem compor uma função de custo para o treinamento de redes neurais convolucionais. Para tal, a utilização de matrizes de distâncias que reúne distâncias de n amostras foi considerada, não se limitando a pares como na *contrastive loss* ou triplas como na *triplet loss*. Considerou-se também a ideia de inserir um termo de regularização, como algumas propostas fazem com a *softmax loss*, porém este é inserido na própria matriz de distâncias. Considerou-se também, a combinação de características obtidas em treinamentos distintos, variando em cada um o parâmetro de regularização.

1.2 Proposta

Partiu-se do problema de criar um espaço de separação que minimiza a distância intraclasse e maximiza a distância interclasse das características obtidas por uma rede neural convolucional, e apresenta-se aqui uma função de custo baseada em matrizes de distâncias que tem as seguintes características:

- Realiza o treinamento em duas etapas. Os filtros convolucionais extraem as características (*deep features*) porque considera-se que a etapa de extração de características deva ter penalidades diferentes da etapa de classificação. O classificador realiza somente a separação e não interfere na extração de características. Assim, existe o benefício do uso de qualquer classificador.
- Utiliza matrizes de distâncias. Distâncias entre amostras é um recurso utilizado em algumas funções de custo, entretanto nesta tese utilizou-se esta abordagem com matrizes de distâncias que induzem uma distribuição espacial das

amostras. No trabalho de [Queiroz et al. \(2009\)](#) é apresentada uma métrica para avaliação de agrupamentos baseada em matrizes de distâncias, onde é possível verificar a densidade de cada agrupamento pelas semelhanças das amostras. Na função de custo proposta aqui minimiza-se o erro entre duas matrizes de distâncias: *i*) a primeira gerada pelas características extraídas em um lote de imagens que alimentam uma rede neural convolucional e *ii*) a gerada a partir dos respectivos rótulos do lote de imagens. Desta forma, trabalha-se no espaço projetado pelas saídas das camadas convolucionais.

- Regularização na matriz de distâncias. Alguns trabalhos utilizaram matrizes de distâncias para auxiliar na tarefa de aprendizagem, como o desenvolvido por [Silvestre et al. \(2015\)](#), onde a matriz é incorporada como informação adicional para regularizar Máquinas de Aprendizado Extremo. Como forma de evitar o sobreajuste do modelo na etapa de extração de características, uma perturbação na matriz de rótulos é realizada para gerar uma incerteza sobre as classes, evitando assim sobreajuste dos parâmetros do modelo.
- Combinação de diferentes soluções. Como feito nos trabalhos de [Yang et al., 2018](#)) e [Xu et al., 2016](#)) pode-se combinar diversas soluções, cada uma com diferente nível de regularização. A combinação apresentada aqui é realizada pela média do conjunto de características obtidas em treinamentos distintos, o que permite trazer uma maior cobertura no espaço de separação e evitar sobreajuste do modelo.

1.3 Objetivos

A apresentação de uma nova função de custo, LDMAT (*Loss Function Based on Distances Matrices*), que utiliza matrizes de distâncias foi o principal objetivo deste trabalho. Para alcançar este fim, as seguintes etapas foram seguidas:

- Conhecer e estudar as arquiteturas amplamente utilizadas de redes neurais convolucionais.
- Analisar as funções de custo propostas na literatura para o treinamento de CNNs.
- Implementar a função de custo LDMAT e realizar testes em conjunto de dados utilizados na tarefa de classificação de imagens.
- Desenvolver um protocolo para o treinamento de uma rede neural convolucional com a função de custo LDMAT.

- Analisar os diversos aspectos de regularização no treinamento com a função de custo LDMAT.

1.4 Contribuições

Considera-se que a função LDMAT apresenta as seguintes contribuições para a tarefa de classificação de imagens com redes neurais convolucionais porque:

- utiliza a técnica de regularização na própria função de custo;
- permite a realização de combinação de soluções, que melhora o desempenho em relação a abordagem sem combinação;
- é uma função de custo independente da *softmax loss* para treinar CNNs;
- realiza o treinamento em duas etapas separando a etapa de extração de características da etapa de classificação, o que permite utilizar qualquer classificador na segunda etapa e
- apresenta resultados superiores a *softmax loss* nos conjuntos de dados testados e a diversas funções de custo com resultados publicados.

Capítulo 2

Arquiteturas de Redes Neurais Convolucionais

Assim como no modelo de neurônio artificial proposto por McCulloch & Pitts (1943), a motivação para construir estruturas profundas em camadas tem inspiração no cérebro de mamíferos. De acordo com Hubel & Wiesel (1959), a área visual do cérebro é composta por células simples e células complexas. Enquanto células simples realizam a extração de características, as células complexas combinam diversas características locais de uma pequena vizinhança espacial (Siagian & Itti, 2007) (Riesenhuber & Poggio, 1999). Estas descobertas serviram de inspiração para muitos modelos de aprendizado de máquina, entre eles as redes neurais convolucionais.

2.1 Histórico das Redes Neurais Convolucionais

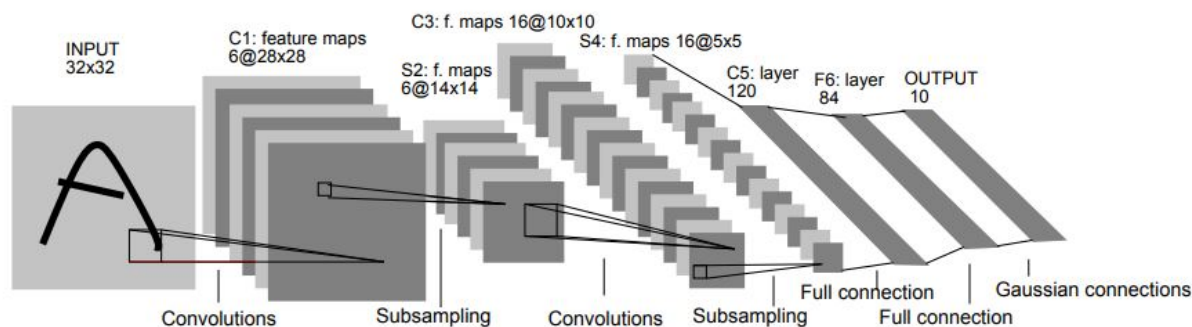
Alguns consideram que o primeiro modelo de aprendizado profundo foi desenvolvido pelo *Group Method of Data Handling* (GMDH), que treinaram um modelo baseado em redes *perceptron* multicamadas. Ivakhnenko (1971) descreveu a rede GMDH com 8 camadas, esta rede profunda é incrementalmente treinada e podada, entretanto as unidades da rede usavam função de ativação polinomial, que implementa as polinomiais de Kolmogorov-Gabor para introduzir não-linearidade, técnica bem diferente das funções de ativação utilizadas atualmente (Ivakhnenko, 1971).

Em 1975, Fukushima apresentou a rede Cognitron (Fukushima, 1975) que é uma adaptação da rede *Perceptron*, desenvolvida por Rosenblatt (1958) e foi a primeira rede a incorporar ideias da neurofisiológicas sobre o córtex visual descobertas nos anos de 1960 por Hubel & Wiesel (1959) em um *framework* de aprendizagem. A rede Cognitron era capaz de reconhecer padrões em imagens, porém não era robusta devido às variações na entrada. Em 1980, Fukushima desenvolveu a rede Neocognitron, uma extensão da Cognitron com aprendizado não-supervisionado, multicamadas, e possuía

uma certa robustez a variações nos padrões de entrada (Fukushima, 1980). Neocognitron introduziu as camadas convolucionais compostas de um conjunto de operações de convolução parametrizadas com um peso, sob a forma de matriz retangular, que são duplicadas sobre a entrada 2D através de um processo de deslocamento. O mesmo modelo introduziu a subamostragem para promover um certa tolerância para pequenos deslocamentos em uma imagem de entrada.

ConvNet é a primeira rede neural convolutiva com a topologia utilizada atualmente, esta foi descrita por LeCun *et al.* (1989). A diferença da abordagem apresentada pela ConvNet é o treinamento supervisionado usando o algoritmo *backpropagation* para o ajuste dos parâmetros. Em 1998, a rede ConvNet foi melhorada e aplicada para o reconhecimento de caracteres (Lecun *et al.*, 1998). Esta arquitetura, nomeada LeNet-5¹(Figura 2.1), realizava a extração de características de forma hierárquica e apresentava tolerância a um certo nível de rotações nas imagens de entrada.

Figura 2.1: Arquitetura de rede LeNet-5. Fonte: Lecun *et al.* (1998)



2.2 Funcionamento da Camada Convolutiva

As redes neurais convolucionais, em linhas gerais, seguem os mesmos princípios das redes neurais artificiais de propósito geral. Existem algumas diferenças que são as operações de convolução e subamostragem. A ideia de usar convolução com pesos fixos para extrair características já era amplamente utilizada, porém nas CNNs os pesos das máscaras de convolução são obtidos no processo de treinamento da rede (Lecun *et al.*, 1998). As convoluções operam em dados de mesmo domínio, por exemplo, nas imagens os dados de entrada são *pixels* que contêm uma correspondência espacial na imagem. Como a convolução utiliza as mesmas máscaras em um processo de janela deslizante, os pesos são compartilhados com toda a imagem de entrada, o que possibilita realizar mais operações com os dados utilizando os mesmos parâmetros. As operações de convolução tornam possíveis melhores representações das camadas

¹yann.lecun.com/exdb/lenet

e torna mais intuitivo o uso de camadas empilhadas em profundidade. Apesar de compartilhar os pesos e possuir menos parâmetros em relação às redes neurais de propósito geral, o processo de convolução requer grande quantidade de operações de multiplicação e soma, tornando o treinamento custoso computacionalmente.

A seguir apresentam-se as equações da operação de convolução utilizada na etapa de alimentação de uma CNN, assim como a formulação utilizada no ajuste dos parâmetros na retropropagação do erro.

As primeiras camadas de um bloco convolutiva realizam operações de convolução. Considera-se que cada camada l possui I^l canais de entrada e O^l canais de saída, onde $q = 1, \dots, O^l$, os canais de saída também são chamados de mapas de características. Se considerarmos uma imagem em cores no formato RGB a entrada é uma imagem com $M \times N$ dimensões e 3 canais de profundidade. Na Equação 2.1 é exibido o cálculo das operações em uma camada convolutiva (Zhang, 2016).

$$C_q^l = \sigma \left(\sum_{p=1}^{N_{input}^l} C_p^{l-1} * k_{p,q}^l + b_q^l \right) \quad (2.1)$$

Além das operações de convolução, a camada convolutiva soma um termo de viés (b^l) em cada mapa de característica da camada. A última operação da camada convolutiva é uma função de ativação σ que insere uma não-linearidade aos dados (Zhang, 2016). O operador $*$ representa a operação de convolução, que é apresentada em detalhes na Equação 2.2,

$$C_q^l(i, j) = \sigma \left(\sum_{p=1}^{N_{input}^l} \sum_{u=-r/2}^{r/2} \sum_{v=-c/2}^{c/2} C_p^{l-1}(i-u, j-v) \cdot k_{p,q}^l + b_q^l \right) \quad (2.2)$$

onde, r e c representam a largura e comprimento das máscaras de convolução. Observe que a convolução é computada em volumes tridimensionais, ou seja, a imagem de entrada tem I canais de entrada e estes valores são somados na terceira dimensão para formar cada canal de saída. Para computar estas operações as máscaras de convolução k possuem 4 dimensões em cada camada, sendo: $r \times c \times I \times O$.

Para realizar a retropropagação do erro na camada de convolução, a função de ativação necessita ser multiplicar pela sua derivada, σ' , como mostrado na Equação 2.3.

$$\nabla C_{q,\sigma}^l = \nabla C_q^l \cdot \sigma' \left(C_q^l \right) \quad (2.3)$$

A estimativa do gradiente da máscara de convolução k^l é feita como apresentada na Equação 2.4. Observe que se faz o uso da operação de convolução do erro propagado com o mapa de características de entrada. Observa-se que é necessário um ajuste para que a convolução aconteça no sentido contrário da operação realizada no

momento da fase de alimentação da rede. Para que isso aconteça, faz-se uma rotação 180° no mapa de características, indicado pelo termo C_{rot180} (Zhang, 2016).

$$\nabla k_{p,q}^l = C_{p,rot180}^{l-1} * \nabla C_{q,\sigma}^l \quad (2.4)$$

O erro é retropropagado para a camada anterior (Equação 2.5), nesta operação as máscaras de convolução k^l também são rotacionadas em 180° .

$$\nabla C_p^{l-1} = \sum_{q=1}^{N_{outup}^l} \nabla C_{q,\sigma}^l * k_{p,q,rot180}^l \quad (2.5)$$

Se a operação de convolução utilizada for a padrão, onde a imagem de saída tem tamanho menor do que a de entrada é necessário realizar uma adição de bordas com valores 0 na matriz de erro propagada ∇C_σ^l . A estimativa do gradiente para o termo de viés da camada convolucional é dada pela Equação 2.6, onde M e N são as dimensões do mapa de características na camada l .

$$\nabla b_q^l = \sum_{i=1}^M \sum_{j=1}^N \nabla C_{q,\sigma}^l(i,j) \quad (2.6)$$

2.3 Arquiteturas de redes neurais convolucionais

Nos anos seguintes a 2012, o desafio ILSVRC se tornou referência para testar as inovações nas arquiteturas de CNNs. Isto tornou o conjunto de dados utilizado no desafio muito popular na comunidade científica. As primeiras modificações feitas na rede AlexNet abordaram a profundidade da rede e as dimensões dos filtros de convolução, referidas aqui como abordagens de exploração espacial.

A rede AlexNet, proposta por Krizhevsky *et al.* (2012), faz o uso de 7 camadas, sendo 2 camadas totalmente conectadas com 4096 neurônios e uma camada de saída. A rede LeNet-5 utilizava somente 5 camadas de profundidade. A rede AlexNet também trocou a função sigmoideal, adotada na rede LeNet-5 pela função de ativação ReLU (*Rectified Linear Units*), como mostrado na Figura 2.2. Outra modificação em relação á LeNet-5 foi substituir a subamostragem de média por subamostragem por máximo.

Para evitar sobreajuste dos parâmetros da CNN, a rede AlexNet utilizou a técnica de desligamento de neurônios (*dropout*), como mostrado na Figura 2.3 (Hinton *et al.*, 2012). A ideia da técnica é desligar aleatoriamente algumas conexão da rede e pela capacidade prevenir sobreajuste dos parâmetros, tornando-se popular em arquiteturas de CNNs.

Figura 2.2: Funções de ativação: Sigmoid e ReLU. Fonte: *Activation Functions in Neural Networks*

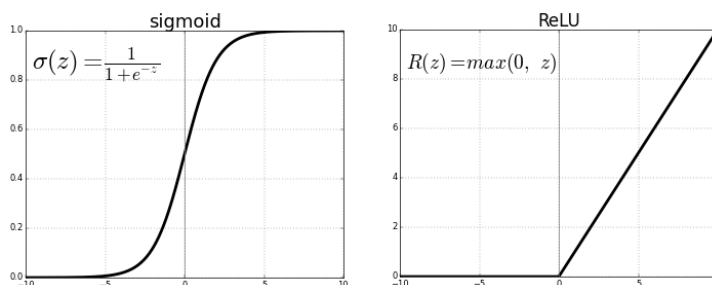
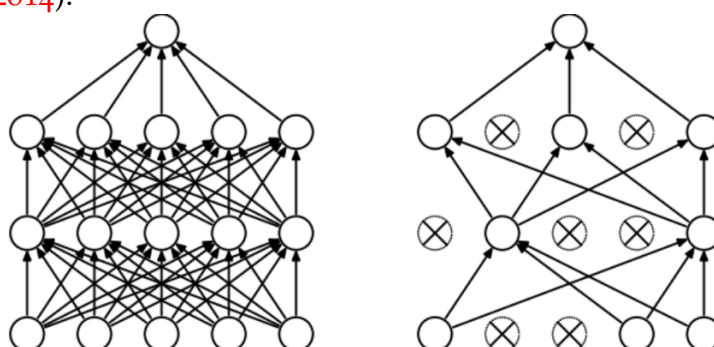


Figura 2.3: Regularização com desligamento aleatório de conexões (*dropout*), fonte: *Srivastava et al. (2014)*.

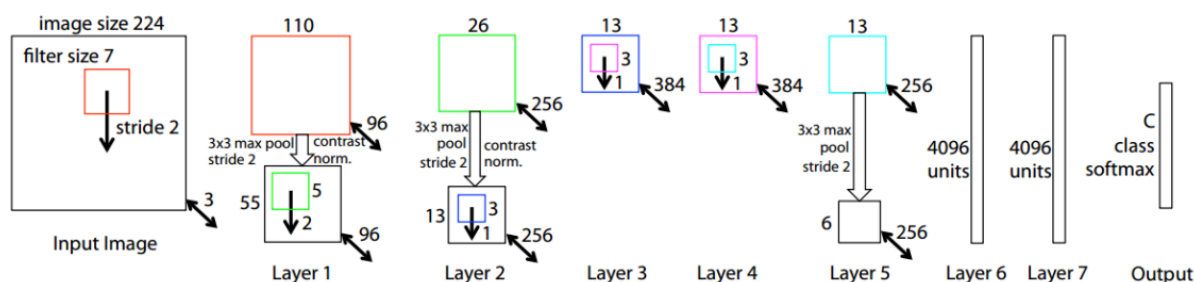


2.3.1 Abordagens de exploração espacial

A primeira geração das CNNs focaram em realizar uma exploração espacial nos projetos das arquiteturas. Esta exploração inclui o tamanho dos filtros de convolução e seu número em cada camada. Além disso, são propostas arquiteturas que aprofundam o número de camadas e apresentam o conceito de bloco de convolução (*Gu et al., 2017*).

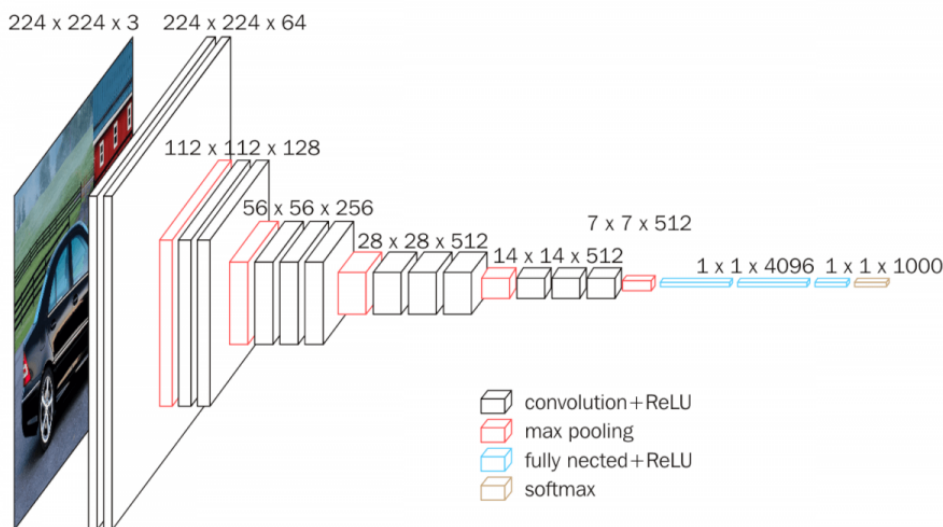
Zeiler & Fergus (2014) com a rede ZefNet, venceram o desafio ILSVRC do ano de 2013 propondo alguns refinamentos na AlexNet. A rede proposta foi desenvolvida a partir de uma análise visual do desempenho da rede AlexNet. Foi monitorado o aprendizado durante a etapa de treinamento e identificou-se alguns problemas no modelo. Este monitoramento deu uma interpretação a nível de neurônio das características internas em cada camada. Foi verificado que poucos neurônios foram ativados na primeira e segunda camada da rede. Além disso, mostrou-se que as características extraídas pela segunda camada apresentavam artefatos de serrilhamento. Baseado nestas observações, a topologia da CNN foi ajustada com melhoria nos hiperparâmetros. Reduziu-se o tamanho dos filtros e o passo de convolução (*stride*) para reter o máximo de características nas duas primeiras camadas convolucionais. O projeto da arquitetura da rede ZefNet é mostrado na Figura 2.4.

Figura 2.4: ZefNet. Fonte: Zeiler & Fergus 2014



Simonyan & Zisserman (2015) propuseram a arquitetura chamada de VGG, em homenagem ao grupo de pesquisa *Visual Geometry Group* da Universidade de Oxford, ao qual participou da competição. Apesar da VGG ter perdido o primeiro lugar na competição ILSVRC de 2014, a rede ficou conhecida devido a sua simplicidade, homogeneidade topológica e aumento de profundidade. VGG foi projetada com 19 camadas (Figura 2.5) sendo mais profunda que AlexNet e ZefNet (Gu et al., 2017).

Figura 2.5: Arquitetura da rede VGG-16. Fonte: VGG in TensorFlow

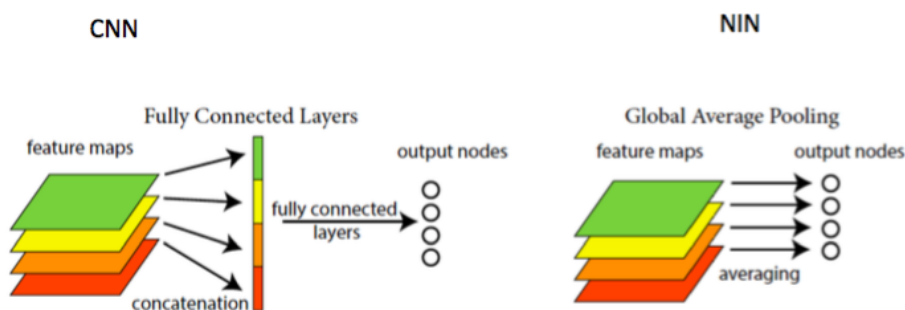


Baseado nas descobertas feitas pela ZefNet de que filtros de tamanhos pequenos podem melhorar o desempenho das CNNs, a rede VGG substituiu os filtros de convolução de dimensão 11x11 e 5x5 por camadas com filtros 3x3. Foi demonstrado experimentalmente que o uso concorrente de filtros 3x3 pode induzir o efeito de filtros mais largos. Apesar do uso de pequenos filtros, a rede VGG sofre com o custo computacional alto.

No trabalho de Lin et al. (2013) foi proposta a substituição das camadas de convolução por uma outra rede: a *Network in Nerwork*. Esta é uma rede de estrutura geral

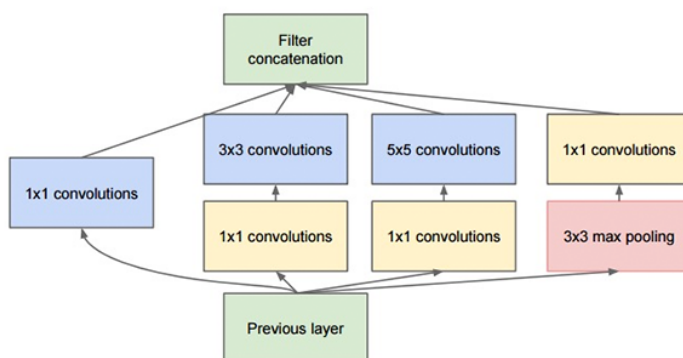
que pode ser computada mais rapidamente do que as operações de convolução. Ela substitui os filtros na camada convolucional por uma micro-rede, como as redes rede *perceptron* multicamadas. Esta micro-rede pode ser entendida como uma convolução com máscara de tamanho 1×1 . Os mapas de características são obtidos pelo deslizamento da micro-rede sobre a entrada. Antes da camada de concatenação foi utilizada a subamostragem de média global (*global average pooling*) sobre o mapa de características. Esta técnica (exibida na Figura 2.6), reduz o número de características na última camada trazendo uma perda de informação que pode dificultar a capacidade de generalização da rede.

Figura 2.6: Subamostragem de média global. Fonte: *Global Average Pooling*



Szegedy et al. (2015) com a rede GoogleNet venceram a competição ILSVRC de 2014. Esta rede também é conhecida como *Inception-V1* porque introduziu um conceito novo de bloco: o módulo *inception*, como mostrado na Figura 2.7. Este bloco incorpora uma transformada convolucional multi-escala usando: divisão, transformação e junção para a extração de características.

Figura 2.7: Bloco *Inception*. Fonte: *Szegedy et al. (2015)*



Na rede GoogleNet as camadas convolucionais convencionais são substituídas por pequenos blocos, de forma similar a ideia de substituir cada camada com uma micro-rede como proposto na arquitetura *Network in Network*. Este módulo *inception* utiliza filtros de diferentes tamanhos: 1×1 , 3×3 e 5×5 . Desta forma, captura-se a informação espacial em resoluções diferentes. A exploração da ideia de dividir, transformar

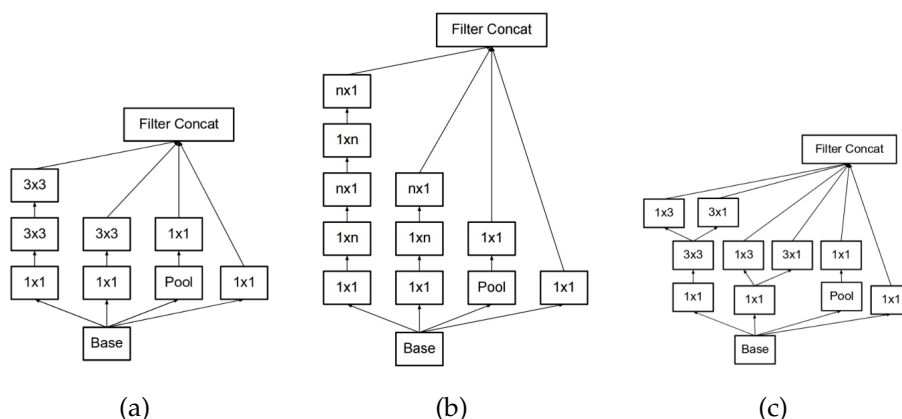


Figura 2.8: Variações do bloco *Inception*. Fonte: Szegedy et al. (2016)

e juntar feita pela rede GoogleNet, ajudou a resolver o problema da diversidade nas variações de imagens dentro da mesma classe. GoogleNet regula a computação pela adição de uma camada de afinamento usando filtro de convolução 1×1 . Na rede nem todos os canais de saída são conectados em todos os canais de entrada seguintes, desta forma supera-se o problema de redundância de informação. Com isto também reduz-se o custo computacional omitindo canais que não são relevantes (Gu et al., 2017).

Outro fator importante foi a regularização feita com a normalização de lotes (Ioffe & Szegedy, 2015). A normalização de lotes unifica a distribuição do mapa de características normalizando os valores para terem média zero variância unitária. A rede GoogleNet também faz o uso do otimizador RMSprop e introduz o conceito de classificadores auxiliares para a aumentar a taxa de convergência (Tieleman & Hinton, 2012). Entretanto, a principal desvantagem da rede GoogleNet em relação à VGG é a sua heterogeneidade de topologia.

As redes *Inception-V2* e *Inception-V3* foram apresentadas no mesmo artigo de Szegedy et al. (2016). Estas arquiteturas propõem mudanças no módulo *inception* introduzindo a fatoração de filtros. Desta forma, substitui-se um filtro de tamanho 5×5 por duas operações com filtros de tamanho 3×3 , como ilustrado na Figura 2.8(a). Convoluções com filtros de tamanho 5×5 são 2.78 vezes mais custosas que operações 3×3 (Gu et al., 2017).

Além disso, também foram fatorados os filtros de convolução em $n \times n$ dimensões para combiná-los com convoluções $1 \times n$ e $n \times 1$, como exibido na Figura 2.8(b). Uma convolução 3×3 é equivalente a realizar primeiro uma convolução com um filtro 1×3 e então uma convolução com outro filtro 3×1 em seguida. Esta proposta do método é 33% por cento mais barato computacionalmente que uma única convolução 3×3 .

O banco de filtros no módulo também foi expandido, ficando mais largo ao invés de profundo, como mostrado na Figura 2.8(c). Esta expansão foi feita para remover a representação de gargalo. Se o módulo for feito mais profundo, isto iria ter uma excessiva redução de dimensões e perda de informação.

Na proposta da rede *Inception-V3*, foi verificado que os classificadores auxiliares não contribuem muito até perto do fim do treinamento, quando a taxa de acurácia está perto da saturação. Para tentar resolver este problema foi feito o uso de normalização de pesos nos classificadores auxiliares. Para a generalização foi feita uma suavização de rótulos para prevenir sobreajuste.

Os classificadores auxiliares trazem a informação para a generalização do modelo. Este é um tipo de componente adicionado para a função de custo que previne que a rede fique muito confiante sobre a classe. Na proposta da função de custo com matrizes de distâncias fizemos uma proposta de técnica de regularização semelhante. A diferença principal é que realizamos uma perturbação diretamente na matriz de distâncias. Caso fosse feita uma suavização nos rótulos o efeito seria mascarado no cálculo da distância entre as características extraídas (Gu *et al.*, 2017).

2.3.2 Exploração da profundidade

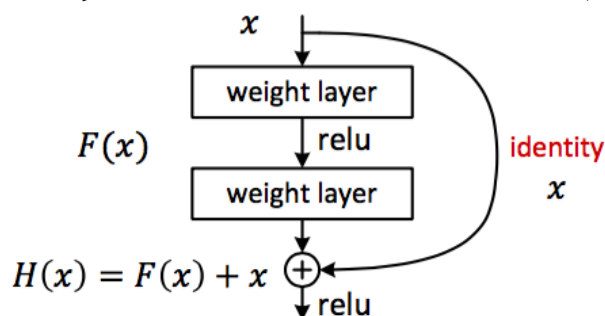
Um dos problemas das redes profundas é que se tem um treinamento lento e rápida convergência quando comparado com as redes rasas. O aumento na profundidade de uma rede melhora o desempenho principalmente para problemas complexos, mas também dificulta o treinamento da rede. Em redes mais profundas, devido a um grande número de camadas, a retropropagação de erro pode resultar em pequenos valores de gradiente nas camadas inferiores. A linha de arquiteturas que exploram múltiplos caminhos do fluxo da informação em profundidade tentam resolver este problema (Gu *et al.*, 2017).

Baseado na ideia que a capacidade de aprendizagem pode ser melhorada pela variação da profundidade da rede, Srivastava *et al.* (2015) propuseram a rede de rodovias. A rede proposta explora a profundidade para melhorar o aprendizado de característica usando um cruzamento de camadas para o treinamento. Esta rede com 50 camadas mostrou uma taxa de convergência melhor que outras redes profundas na base de dados ImageNet. Foi mostrado experimentalmente que o desempenho de uma rede plana cai depois de adicionar mais que 10 camadas escondidas (Glorot & Bengio, 2010). Na rede de rodovias, o fluxo desimpedido de informações através das camadas é ativado ao transmitir duas unidades de restrição dentro de uma camada. A agregação de informações na camada atual e as informações das camadas anteriores criam um efeito regularizador, facilitando o treinamento baseado em gradiente

de redes muito profundas. Isso permite o treinamento de uma rede com mais de 100 camadas com o algoritmo do gradiente descendente estocástico.

He *et al.* (2016) propuseram uma CNN com 152 camadas de profundidade, a qual venceu a competição ILSVRC de 2015. A rede chamada de ResNet introduziu uma metodologia para treinamento de redes mais profundas, que possui uma complexidade computacional menor do que as redes propostas anteriormente.

Figura 2.9: Bloco Residual. Fonte: He *et al.* (2016)



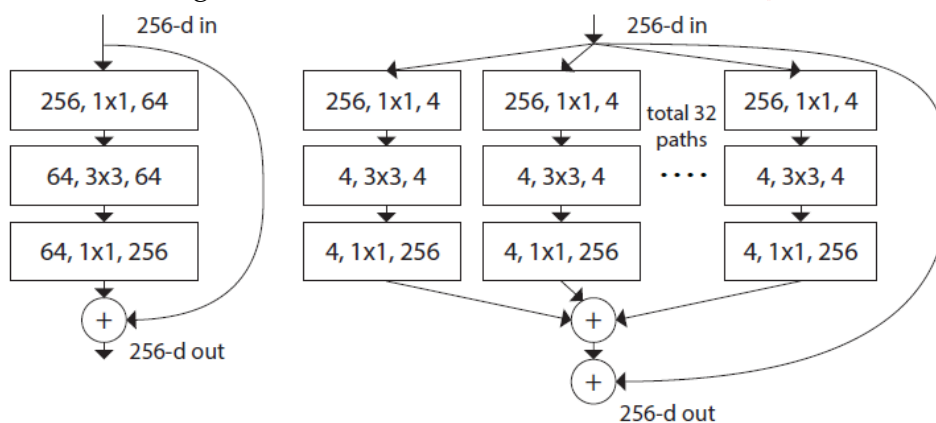
A rede ResNet introduziu atalhos de conexões nas camadas para permitir a conectividade entrecamadas, como mostrado na Figura 2.9. Porém estas conexões são independentes dos dados e sem parâmetros em comparação com as redes de rodovias. Nas redes de rodovias, quando um atalho é criado, as camadas representam funções não-residuais. No entanto, na rede ResNet, as informações residuais sempre são passadas e os atalhos de identidade nunca são fechados. As ligações residuais (conexões de atalho) aceleram a convergência das redes com muitas camadas, evitando problemas de diminuição do gradiente. Mesmo com maior profundidade, a rede ResNet possui menor complexidade computacional que a rede VGG. (Gu *et al.*, 2017)

Szegedy *et al.* (2017) introduziram no mesmo artigo as redes *Inception-V4* e *Inception-Resnet*. A ideia foi projetar blocos mais uniformes, porque verificou-se que alguns dos blocos *inception* eram mais complexos do que o necessário. Isto permitiu aumentar o desempenho adicionando mais blocos. *Inception-v4* também introduziu o bloco especializado chamado bloco de redução, que é utilizado para mudar a largura e comprimento dos mapas de características. Inspirado pelo desempenho da rede ResNet, um módulo *Inception* híbrido foi proposto. Foram introduzidas as conexões residuais que adicionam a saída de uma operação de convolução em um módulo *Inception* para a entrada. Para adição residual do trabalho, a entrada e a saída depois da convolução deve ter as mesmas dimensões. Consequentemente, são utilizadas convoluções 1×1 depois das convoluções originais, para não se ter diferenças de tamanhos de profundidade.

ResNext, também conhecida como rede de transformação residual agregada, é uma melhora das redes *Inception* (Xie *et al.*, 2017), porém também tem características

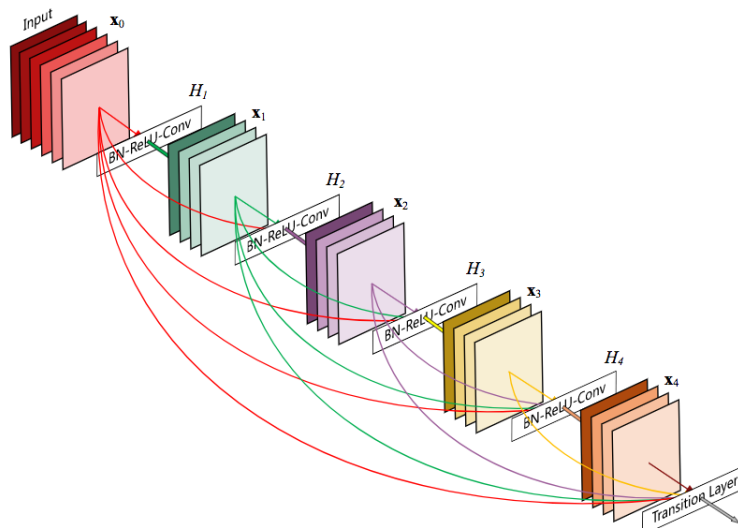
das redes VGG e ResNet. Esta rede explora a topologia de divisão, transformação e junção pela introdução do termo de cardinalidade. A cardinalidade é uma dimensão adicional, na qual se refere ao tamanho do conjunto de transformações, como mostrado na figura 2.10. ResNext utilizou uma topologia homogênea profunda e simplificou a arquitetura GoogLeNet pela fixação espacial para filtros de 3×3 com blocos de divisão, transformação e junção (Gu *et al.*, 2017).

Figura 2.10: ResNext. Fonte: Xie *et al.* (2017)



As redes DenseNets utilizam múltiplos caminhos para o fluxo da informação ao longo da rede. Esta rede é baseada nas Redes de rodovias e na ResNet. A rede também foi proposta para resolver o problema do desaparecimento do gradiente (Huang *et al.*, 2017). Um dos problemas identificados com a rede ResNet é que esta preserva informação através de adição de mapas de características nos quais alguns contribuem muito pouco para o aprendizado. Além disso, ResNet tem um número grande de pesos, com cada camada tendo um conjunto de pesos separados.

Figura 2.11: DenseNets. Fonte: Huang *et al.* (2017)



Para resolver estes problemas na rede Resnet, a arquitetura da DenseNet usou conectividade de cruzamento de camadas mas de uma forma modificada. Mapas de características de todas as camadas precedentes são usadas como entrada nas camadas seguintes, como mostra a Figura 2.11. Como a DenseNet concatena características das camadas anteriores ao invés de adicioná-las, a rede ganha potencialidade diferente entre informação que é adicionada para a rede e informação que é preservada. A rede DenseNet tem restringido a estrutura das camadas, entretanto, isto torna a computacionalmente cara com o aumento do número do mapa de características. A admissão direta de cada camada para o gradiente através da função de custo melhora o fluxo de informação ao longo da rede. Isto incorpora um efeito de regularização, com a redução de sobreajuste nas tarefas com pequenos conjuntos de treinamento (Gu *et al.*, 2017).

2.3.3 Abordagens com múltiplas conexões baseadas na largura

O empilhamento de várias camadas permite aprender diversas representações de características, mas não necessariamente aumenta o poder de aprendizado da rede. Um problema das arquiteturas profundas é que algumas camadas podem não aprender características úteis. Para resolver esse problema, o foco de linhas de pesquisas foi voltado para arquiteturas rasas e amplas ou invés de arquiteturas profundas e estreitas (Gu *et al.*, 2017).

A principal desvantagem associada a redes residuais é o problema do reuso de características, onde algumas transformações de características contribuem muito pouco para a aprendizagem. Este problema foi abordado pela rede WideResNet (Zagoruyko & Komodakis, 2016). Foi identificado que o principal potencial de aprendizagem em residuais está nos blocos residuais. A profundidade teria um efeito suplementar.

A rede residual mais ampla foi baseada na observação de que quase todas as arquiteturas anteriores às redes residuais, como a VGG, eram mais amplas quando comparadas à ResNet. A rede WideResNet explorou o poder dos blocos residuais criando uma ResNet mais larga do que profunda. Aumentou-se a largura introduzindo um fator adicional que controla a largura da rede. Mostrou-se que com o alargamento das camadas é fornecida uma maneira mais eficaz de melhorar o desempenho do que aprofundar redes residuais. Isto mostrou que a rede WideResNet pode treinar de uma maneira melhor do que redes profundas. Porém, a rede WideResNet tem o dobro do número de parâmetros em comparação com o ResNet (Gu *et al.*, 2017).

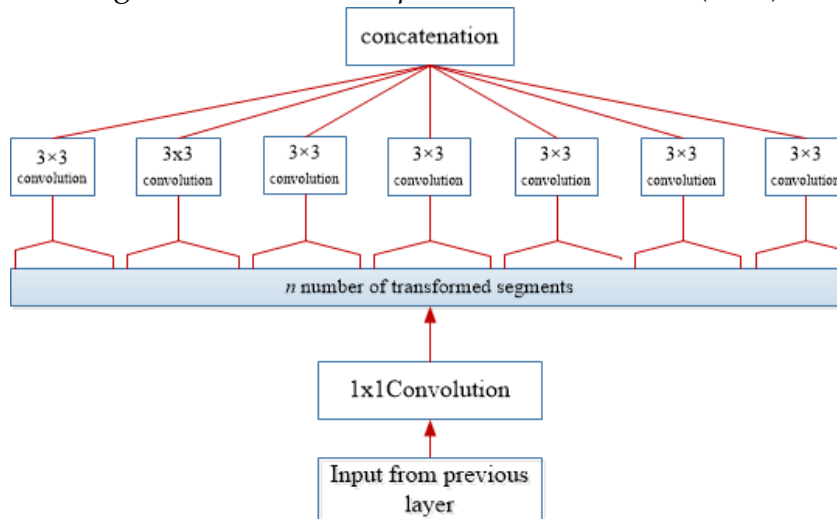
O aumento drástico na profundidade do mapa de características com a perda de informações espaciais interfere na capacidade de aprendizado. A rede ResNet mostrou bons resultados para o problema de classificação de imagens. No entanto, na rede ResNet, a exclusão do bloco convolucional, onde a dimensão espacial e o canal

mudam, geralmente resulta em deterioração do desempenho do classificador. Nesse sentido, a rede ResNet melhorou o desempenho, reduzindo a perda de informações associada à queda da unidade residual. Para abordar o problema de interferência na aprendizagem da ResNet, a rede Piramidal foi proposta (Han *et al.*, 2017).

Ao contrário da diminuição na largura espacial com o aumento da profundidade da ResNet, a rede piramidal aumentou a largura da rede gradualmente por unidade residual, em vez de manter a mesma dimensão espacial dentro de cada bloco residual. Conexões residuais foram inseridas entre as camadas usando o mapeamento de identidade preenchido com zero. A vantagem é que este mapeamento precisa de menos número de parâmetros, o que resulta em melhor generalização. A rede piramidal usa duas abordagens diferentes para o alargamento da rede, incluindo adição e multiplicação do alargamento. A diferença entre os dois tipos de alargamento é que a estrutura piramidal aumenta linearmente e a outra aumenta geometricamente (Gu *et al.*, 2017).

A rede *Xception* é uma arquitetura *Inception* externa, que explora a ideia de convolução separável em profundidade. Esta rede foi proposta por Chollet (2016), o criador e mantenedor da biblioteca Keras¹.

Figura 2.12: Bloco *Xception*. Fonte: Chollet (2016)



A rede *Xception* modificou o bloco *inception* original, ampliando-o e substituindo dimensões espaciais de 1×1 , 5×5 e 3×3 por uma única dimensão (1×1 seguida por 3×3), como mostrado na Figura 2.12. A rede *Xception* é mais eficiente em termos computacionais, separando a correlação espacial com a de canal. A rede funciona mapeando primeiramente a saída convolvida para espaços de baixa dimensão usando convolução 1×1 e depois realiza uma transformação espacial. Na rede *Xception*, k é uma cardinalidade que define a largura e que determina o número de transformações.

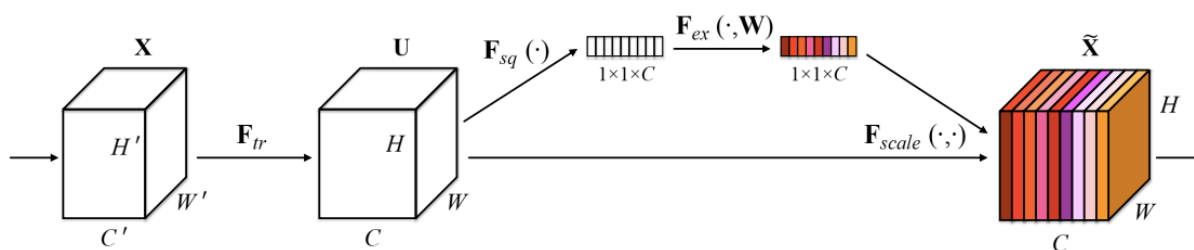
¹<http://keras.io>

A rede *Xception* facilita a computação, convoluindo cada canal separadamente através de eixos espaciais, que são seguidos por convolução 1×1 para executar a correlação entre canais. Na rede *Xception*, a convolução 1×1 é usada para regular a profundidade do canal. Nas arquiteturas convencionais da CNN, a operação convolucional convencional usa apenas um segmento de transformação, o bloco inicial usa três segmentos de transformação, enquanto que na rede *Xception* o número de segmento de transformação é igual a um número de canais. Embora a estratégia de transformação adotada não reduza o número de parâmetros, ela torna o aprendizado mais eficiente e resulta em melhor desempenho (Gu *et al.*, 2017).

2.3.4 Exploração do mapa de características

Hu *et al.* (2018) apresentaram uma rede que utiliza um bloco novo para a seleção de mapas de características relevantes para a classificação de uma imagem. Este bloco foi chamado de bloco-SE (*squeeze and excitation*) e é mostrado na Figura 2.13. Nele suprime-se os blocos de características menos importantes. O trabalho deste bloco é dividido em duas operações: aperto e excitação. Os filtros convolucionais capturam informações localmente, mas ignoram a correlação das características que são de fora deste campo receptivo. Para se obter uma visão global do mapa de características, os blocos de aperto geram um canal estatístico pela supressão espacial da informação da entrada convolvida. A saída da operação de aperto serve de entrada para a operação de excitação que modela as interdependências por um mecanismo de seleção. A operação de excitação atribui pesos para os mapas de características usando 2 camadas (Gu *et al.*, 2017).

Figura 2.13: Bloco-SE. Fonte: Hu *et al.* (2018)



Hu *et al.* (2018) propuseram o uso de blocos residuais e mapeamento de identidade onde ambos são responsáveis por re-escalar o canal. Foi utilizada a ideia de blocos-SE para calibrar o mapa de características baseada na sua contribuição no discriminador de classe. A principal desvantagem de se utilizar o bloco SE com a ResNet é que esta somente considera a informação residual para determinar o peso de cada canal. Isto minimiza o impacto dos blocos SE e faz a informação da Resnet redundante. Este

problema é resolvido gerando mapas de características a partir de mapeamento de identidade e residual. Neste contexto, representações globais de mapas de características foram geradas pela subamostragem de média global, enquanto que a relevância do mapa de características foi estimado fazendo uma competição entre mapeamento de identidade e mapeamento residual.

2.3.5 Conclusões do capítulo

Com os principais trabalhos analisados aqui neste capítulo verificou-se que existe uma busca por técnicas que melhorem a capacidade de generalização do modelo. Nas propostas de arquiteturas de redes neurais convolucionais um problema recorrente que buscaram tratar é caso do desaparecimento do gradiente que ocorre quando a arquitetura é muito profunda. Alguns pesquisadores seguiram na linha de aumentar a largura das CNNs, ou criar conexões residuais ou cruzadas para suprimir este problemas. Identifica-se que este problema do desaparecimento do gradiente possa ser minimizado com o treinamento em duas etapas com uma função de custo para o treino somente dos filtros de extração de características.

Capítulo 3

Funções de Custo para CNNs

Para guiar o processo de ajuste de parâmetros de uma rede neural artificial utiliza-se uma função de custo para computar a distância entre o valor previsto e o valor real (Li *et al.*, 2021). Muitas funções de custo foram projetadas para o treinamento de modelos de redes neurais como, por exemplo, erro médio absoluto, erro médio quadrático, função de Hinge ou função de entropia cruzada (Rosasco *et al.*, 2004).

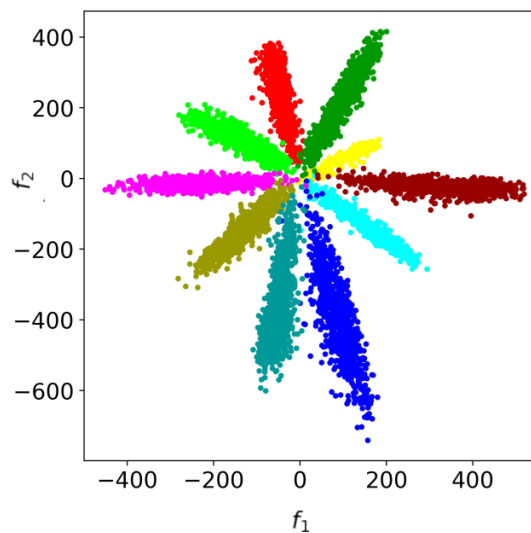
A função de custo entropia cruzada, é uma das funções de custo mais utilizadas para o treinamento de redes neurais convolucionais e é usada para avaliar a diferença entre a distribuição de probabilidade obtida no treinamento atual e a distribuição real. A função tem um intervalo de saída entre 0 e 1 e é baseada na divergência-KL (Kullback-leibler) (Kullback & Leibler, 1951). A computação da função é feita comparando a probabilidade predita com o valor real de saída em cada classe, obtendo assim o valor da penalidade com base nestas distâncias. A função de entropia cruzada utiliza penalidade logarítmica, portanto a função fornece uma pontuação próxima de 0 para diferenças menores e uma pontuação próxima de 1 para diferenças maiores. Esta função é sempre utilizada com a função de ativação *softmax* o que a torna conhecida como *softmax loss* (Rosasco *et al.*, 2004). A Equação 3.1 apresenta a função, onde y representa os rótulos na codificação com somente um índice com valor 1 e os demais com valores 0, e f representa as previsões. Para minimizar a diferença entre as previsões e os valores dos rótulos, a função de custo *softmax* é definida como:

$$\mathcal{L}_{\text{Softmax}} = \frac{1}{N} \sum_i -\log \left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \right) \quad (3.1)$$

A função de custo *softmax loss* é amplamente utilizada em redes neurais convolucionais como, por exemplo, na rede AlexNet (Krizhevsky *et al.*, 2012) e ResNet (He *et al.*, 2016), no entanto, *softmax loss* tem o foco em gerar uma superfície de separação para a classificação e não uma compactação dentro da mesma classe ou uma margem entre as diferentes classes, a superfície gerada tem uma separação angular que pode

ser representada na similaridade do cosseno (Liu *et al.*, 2016). Esta representação angular utiliza coordenadas polares no espaço de projeção gerado pela função de custo *softmax loss* (Peng & Yu, 2021). A Figura 3.1 mostra um exemplo de separação da *softmax loss*, observa-se que existem amostras que a distância intraclasse é maior que a inter-classe, o que dificulta a generalização do modelo treinado.

Figura 3.1: Superfície de separação da *softmax loss* na base de dados MNIST. Para gerar a figura foi utilizada a base de treinamento em uma arquitetura de CNN com camada totalmente conectada com dois neurônios antes da camada de saída. Desta forma é possível ter uma visualização 2D do espaço de características. Fonte: Qin *et al.* (2020).



Algumas funções de custo foram propostas para substituir ou melhorar a *softmax loss* no treinamento de redes neurais convolucionais. Pode-se categorizá-las em três linhas principais: a) funções de custo totalmente independentes da *softmax loss*, b) refinamentos da *softmax loss* e c) combinação da *softmax loss* com outras funções de custo. A seguir são apresentados os principais trabalhos na área de função de custo para redes neurais convolucionais.

3.1 Funções de Custo Independentes da *Softmax Loss*

Para melhorar a separação das características treinadas, algumas funções de custo utilizam a distância entre amostras computando o contraste, como acontece na *contrastive loss* (Hadsell *et al.*, 2006), que é utilizada para aprender um mapeamento que minimiza a distância intraclasse e maximiza a distância interclasse. Na proposta inicial, foi utilizada uma rede neural convolucional siamesa baseada na LeNet-5 para reduzir a dimensionalidade de imagens de entrada. Depois da redução de dimensionalidade,

as duas amostras que são originalmente similares (mesma classe) continuam similares no espaço de características, entretanto as duas amostras que são originalmente diferentes (classes diferentes) continuam diferentes. *Contrastive loss* é utilizada para treinar redes neurais convolucionais para a tarefa de reconhecimento de faces Sun *et al.* (2015). A formulação da *contrastive loss* é mostrada nas Equações 3.2 e 3.3,

$$\mathcal{L}_{Contrastive} = \sum_{i=1}^P L \left[(y, f_1, f_2)^i \right] \quad (3.2)$$

$$L((y, f_1, f_2)^i) = (1 - y)L_S(D^i) + y L_D(D^i) \quad (3.3)$$

onde $(y, f_1, f_2)^i$ é um par de características rotuladas, L_S é a função de custo parcial para um par de características da mesma classe, L_D é a parte da função de custo para um par de característica de classes diferentes e P é o número de pares de treinamento. D_w é a reescrita de $D_w(f_1, f_2)$ que representa a distância Euclideana entre as características f_1 e f_2 . L_S e L_D devem ser projetados de modo que minimizando L resultaria em valores baixos de D para pares semelhantes e valores altos de D para pares diferentes. Uma desvantagem desta função é que requer que as amostras de entrada sejam ordenadas antes do que o contraste possa ser calculado, o que implica um custo computacional adicional neste ordenamento das imagens para o treinamento.

Inspirando-se na *contrastive loss*, Schroff *et al.* (2015) propuseram a *triplet loss* para uma CNN na tarefa de classificação de faces (FaceNet), cujo funcionamento da função é baseado em três imagens: imagem âncora, imagem positiva e imagem negativa. A imagem positiva e a âncora são da mesma classe, entretanto a imagem negativa e âncora são de faces de pessoas diferentes. Minimizar função da *triplet loss* implica minimizar a distância entre a âncora e a positiva, e maximizar a distância entre a âncora e a negativa. *Triplet loss* é usualmente usada com CNNs para classificação de faces humanas, no qual requer que o modelo tenha a habilidade de distinguir diferentes indivíduos. Na Equação 3.4 é exibida a formulação da *triplet loss*,

$$\mathcal{L}_{Triplet} = \sum_{i=1}^T \left[\left\| f_i^a - f_i^p \right\|_2^2 - \left\| f_i^a - f_i^n \right\|_2^2 + \alpha \right] \quad (3.4)$$

onde α é a margem que é forçada entre os pares positivos e negativos, e T é o número de triplas de amostras de treinamento.

Contrastive loss e *triplet loss* foram propostas para aumentar as restrições na projeção das características no espaço de separação. Elas podem treinar facilmente conjuntos de dados em grande escala, entretanto a desvantagem é que muita atenção é dada para as características locais, levando a dificuldades de treinamento e tempo longo de convergência.

3.2 Refinamentos da Softmax Loss

Alguns pesquisadores trabalharam para melhorar a projeção das características geradas pela *softmax loss* tornando mais restritivo o processo de aprendizagem, seja por normalização das características, ou inserindo margem de separação no espaço projetado.

O uso da penalidade L_2 na função de custo *softmax loss* foi proposto por [Ranjan et al. \(2017\)](#). As características obtidas com a função de custo L_2 -*softmax* diminuem a variabilidade angular intraclasse e geram representações de características em cada classe com lóbulos mais finos, uma vez que com a função *softmax* padrão a variância angular intraclasse é grande. A função L_2 -*softmax* também gera magnitude das características menores que a *softmax* padrão, com a função a rede se concentra em trazer as características da mesma classe mais próximas uma das outras e separa as características de diferentes classes no espaço normalizado ou angular. Na Equação 3.5 é exibida a formulação da L_2 -*softmax*,

$$\mathcal{L}_{l_2\text{-softmax}} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{W_{y_i}^T f_i + b_{y_i}}}{\sum_{j=1}^C e^{W_j^T f_i + b_j}} \quad (3.5)$$

onde f são as características, W são os pesos e $\|f_i\|_2 = \alpha$, $\forall_i = 1, 2, \dots, N$.

A função de custo l -*softmax* ([Liu et al., 2016](#)) explora a particularidade da *softmax loss* de ser expressa em termos da similaridade do cosseno para produzir um classificador como margem larga na separação das classes. A função de custo l -*softmax* [Liu et al. \(2016\)](#) introduz um parâmetro de margem m e modifica a superfície de decisão da função de custo *softmax* original, no qual aumenta a dificuldade de aprendizagem pela modificação de $\|W\| \|f\| \cos \theta$ para $\|W\| \|f\| \cos m\theta$, assim atenuando um pouco o problema de sobreajuste e produzindo uma margem de decisão (parâmetro m) para tornar a distribuição mais discriminativa. As Equações 3.6 e 3.7 exibem a formulação da l -*softmax*,

$$\mathcal{L}_{L\text{-softmax}_i} = -\frac{1}{N} \sum_{i=1}^N \log \left(\frac{e^{\|W_{y_i}\| \|f_i\| \psi(\theta_{y_i})}}{e^{\|W_{y_i}\| \|f_i\| \psi(\theta_{y_i})} + \sum_{j \neq y_i} e^{\|W_{y_i}\| \|f_i\| \cos(\theta_j)}} \right) \quad (3.6)$$

$$\psi(\theta) = (-1)^k \cos(m\theta) - 2k, \quad \theta \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m} \right] \quad (3.7)$$

onde $k \in [0, m-1]$, quando m é zero, l -*softmax* se torna idêntica a *softmax loss* original.

L -*softmax* introduz o ângulo de margem com uma separação explícita, porém no trabalho de [Liang et al. \(2017\)](#) apresenta-se uma margem suave de separação. Quando se expande $\cos(m\theta_1)$ em séries de Taylor, m deve ser um inteiro positivo e esta margem não cobre todos os ângulos possíveis. É proposto então projetar uma margem suave

de separação com o seguinte critério: $\mathbf{W}_2^T f \leq \mathbf{W}_1^T f - m < \mathbf{W}_1^T f$, onde \mathbf{W} são os parâmetros da camada anterior a camada de saída.

A função de custo *AM-Softmax* (Wang et al., 2018a) também introduz um parâmetro m na formulação original da *softmax loss*, porém este funciona como uma margem aditiva e não uma margem angular como é na função de custo *l-softmax*. *AM-Softmax* normaliza os pesos da última camada de características de saída para reduzir o impacto de imagens de resoluções diferentes nas base de dados. A formulação da *AM-Softmax* é apresentada na Equação 3.8.

$$\mathcal{L}_{AM-softmax} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{s(\mathbf{W}_{y_i}^T f_i - m)}}{e^{s(\mathbf{W}_{y_i}^T f_i - m)} + \sum_{j=1, j \neq y_i}^c e^{\mathbf{W}_j^T f_i}} \quad (3.8)$$

Na Figura 3.2 são exibidas uma representação do espaço de características das funções de custo que apresentam refinamentos da *softmax loss*.

Figura 3.2: Efeito das funções de custo na distribuição das características no espaço de separação. Em (a) *Softmax loss*, (b) *L2-softmax loss* (Ranjan et al., 2017). (c) *L-Softmax* (Liu et al., 2016) e (d) *AM-Softmax* (Wang et al., 2018a). Figura adaptada de Liang et al. (2020).



No trabalho de Luo et al. (2020) é proposta a função de custo *G-softmax*, onde é feita a reescrita da *softmax loss* assumindo que as características f_i pertencem a distribuição Gaussiana, $f_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. A Equação 3.9 mostra a formulação da *G-softmax*,

$$\mathcal{L}_{G-softmax} = -\sum_{i=1}^m y_i \log \left(\frac{\exp(f_i + \lambda \Phi(f_i, \mu_i, \sigma_i))}{\sum_{j=1}^m \exp(f_j + \lambda \Phi(f_j, \mu_j, \sigma_j))} \right) \quad (3.9)$$

onde Φ é a distribuição acumulada Gaussiana (*Cumulative distribution function - CDF*) e λ é um parâmetro controlando a largura da CDF. Quando $\lambda = 0$, a Equação 3.9 se torna a função de custo *softmax loss* padrão.

3.3 Softmax loss combinada com outras funções

Alguns pesquisadores apresentaram novas funções de custo que utilizaram a *softmax loss* combinada com outras funções com o objetivo de inserir termos que funcionam

como regularização na *softmax loss* para melhorar a compactação intraclasse das características.

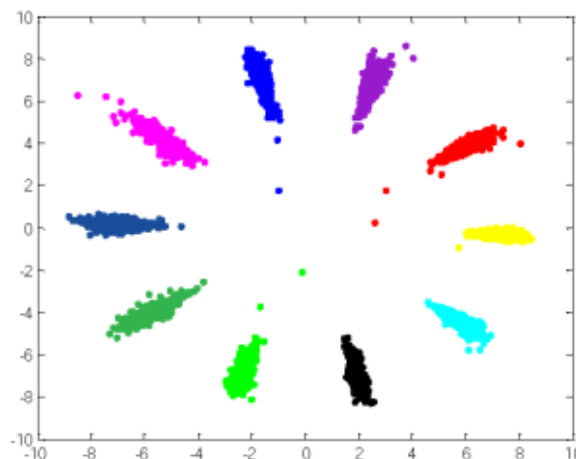
Foi proposto por [Yang et al. \(2018\)](#) o uso de uma CNN para a aprendizagem de protótipos (*generalized convolutional prototype learning - GCPL*), onde as características de cada classe são armazenadas e a classificação é feita, em uma segunda etapa, para a classe que mais se aproxima do padrão projetado. O uso de protótipos segue a mesma ideia da função de custo *contrastive loss*, porém no modelo GCPL foi proposta uma função de custo de protótipo que funciona como uma regularização quando combinada com a *softmax loss*. A função de custo de classificação enfatiza a propriedade de separação e a função de custo de protótipo enfatiza a propriedade de compactação da representação. Combinando estas funções chega-se a representações compactas na mesma classe e separáveis interclasses. Na fase de classificação foi assumido que os protótipos seguem uma distribuição Gaussiana. A formulação da GCPL é apresentada na Equação 3.10

$$\mathcal{L}_{GCPL} = \mathcal{L}_{softmax} + \lambda \left\| f - m_{yj} \right\|_2^2 \quad (3.10)$$

onde o primeiro termo é uma função de classificação que tem objetivo de separar as classes, baseada na entropia cruzada (*softmax loss*), e o segundo termo é o termo de distância baseado no erro médio quadrático, o parâmetro λ realiza o balanceamento das duas funções, m_{yj} é protótipo mais próximo de f da correspondente classe y . A ausência do segundo termo na equação deixa GCPL igual a *softmax loss*.

Na Figura 3.3 é exibida uma superfície de separação da função de custo GCPL.

Figura 3.3: Espaço de separação GCPL gerado no conjunto de dados MNIST, foi utilizado $\lambda = 0.001$. Observe que existe uma melhor compactidade intraclasse, porém o espaço de separação ainda é angular.



A função de custo *center loss* proposta por [Wen et al. \(2016\)](#) combina a *softmax loss* com o erro médio quadrático (*Mean Squared Error - MSE*), onde um centro para cada

classe é aprendido, entretanto, o centro de cada classe é continuamente atualizado durante o processo de treinamento. O propósito da *center loss* é focar na uniformização da distribuição das características da mesma classe. No processo de treinamento os múltiplos centros são simultaneamente atualizados e a distância entre as amostras de entrada e suas correspondentes classes são minimizadas. A Equação 3.11 descreve a *center loss*,

$$\mathcal{L}_{Center} = \mathcal{L}_{Softmax} + \frac{\lambda}{2} \sum_{i=1}^m \|f_i - c_{y_i}\|_2^2 \quad (3.11)$$

onde c_{y_i} denota o centro da classe y_i . Se as amostras de entrada pertencerem à mesma classe, a *center loss* pode garantir que suas distâncias espaciais sejam mais próximas do centro da classe, no entanto as características interclasses não são garantidas como bem separadas. A *center loss* requer que todo o conjunto de treinamento seja considerado e os centros devem ser atualizados em cada iteração, o que é custoso devido à natureza e ao tamanho dos problemas comumente resolvidos pelas CNNs.

A função de custo com uso de centróides de classes pré-definidos uniformemente (*Predefined Evenly-Distributed Class Centroids - PEDCC*) (Zhu et al., 2020) apresenta refinamentos para função de custo *center loss*. Centróides de classe uniformemente distribuídos são gerados, substituindo os centros estimados pela *center loss* e o erro médio quadrático é usado para uma maior distância entre a amostra e seu centro de classe correspondente. *PEDCC loss* utiliza a normalização das características extraídas e combina a função de custo com a AM-Softmax (Wang et al., 2018a). Os centróides são fixos e uniformemente distribuídos, assim não são atualizados durante o treinamento. A Equação 3.12 mostra a formulação da função de custo,

$$\mathcal{L}_{PEDCC} = \mathcal{L}_{PEDCC-AM} + \lambda \sqrt{\mathcal{L}_{PEDCC-MSE}} \quad (3.12)$$

onde as distâncias utilizadas na função de *AM-softmax* e MSE são em relação aos centróides propostos $PEDCC_{y_i}$.

Seguindo a mesma ideia de *PEDCC loss*, Shi et al. (2021b) propuseram a função de custo *Constrained Center Loss - CCL* como forma de inserir restrições na *center loss* em uma abordagem nova de atualização dos centros das classes. A função de custo consiste em dois termos: *softmax loss* e CCL, onde a *softmax loss* separa os vetores de características em classes e CCL procura melhorar a compactação intra-classe. Os centros de classes são aprendidos analiticamente com a utilização de todo o conjunto de treinamento e são impostas restrições nas magnitudes dos vetores de características. Para o treinamento, os pesos das conexões da CNN são fixados e os centros das classes são atualizados com base em uma fórmula analítica. Então, atualiza-se os centros das classes e os pesos das conexões com várias iterações nos lotes com o algoritmo SGD.

O recurso de combinar classificadores é utilizado por [Wang et al. \(2018b\)](#) na função de custo *EM-Loss*. A função de custo é composta por dois termos, o primeiro *M-softmax* é baseado na *softmax loss* com um parâmetro de margem e *E-Softmax* que utiliza diversos classificadores fracos para trazer generalização a função de custo. Estes classificadores fracos são escolhidos com o uso de uma métrica de independência estatística das características geradas. O critério de independência de Hilbert-Schmidt é utilizado para medir esta independência das soluções geradas pelo termo de generalização da função de custo, buscando maior diversidade nas soluções. Este critério mede a dependência mapeando variáveis em um espaço de Hilbert do *kernel* de reprodução, de modo que a dependência não-linear possa ser tratada. Este é dado por $HSIC(Z, F, G) = (n - 1) - 2tr(\mathbf{K}_1 \mathbf{H} \mathbf{K}_2 \mathbf{H})$, onde Z são as amostras com os rótulos. K_1 e K_2 são as funções de *kernel* definidas no espaço F e G respectivamente, $H = \mathbf{I} - n^{-1} \mathbf{1} \mathbf{1}^T$ centra a matriz de Gram para média zero. Alta independência de dois classificadores W_v e W_u significa alta diversidade entre eles. A formulação da função de custo *EM-loss* é dada pela Equação 3.13.

$$\mathcal{L}_{EM} = \alpha_1 \left(-\frac{1}{N} \sum_{i=1}^N y_i \log \hat{y}_i \right) + (\lambda \mathbf{W} \mathbf{H} \mathbf{W}) \quad (3.13)$$

No estágio de teste, é construído o classificador final calculando a média de classificadores fracos

A função de custo de separabilidade de compacidade (*Separability and Compactness Network - SCNet*) ([Zhou et al., 2019](#)) é proposta para treinar uma rede neural convolucional siamesa. SCNet explora a ideia de separabilidade interclasse e compactação intraclasse usando uma rede de canal duplo, onde as amostras são divididas em dois subconjuntos A_{part1} e A_{part2} que são processados em dois canais separados, e em seguida, dois conjuntos de características são obtidos e usados para calcular a função de compacidade. A função de custo final é obtida pela combinação linear da *softmax loss* e a função de compacidade. A principal inovação é o uso da rede siamesa e do critério de distribuição das imagens ao longo da rede, porém, por ser uma rede de canal duplo é preciso treinar o dobro de parâmetros que uma rede canal simples. A Equação 3.14 apresenta a formulação da SCNet,

$$\mathcal{L}_{SCNet} = \mathcal{L}_{softmax} + \beta \frac{1}{2N} \sum_{i=1}^N N \left\| f_i - f'_i \right\|^2 \quad (3.14)$$

onde β é um termo de ponderação, f_i e f'_i representam as características de cada parte da rede siamesa.

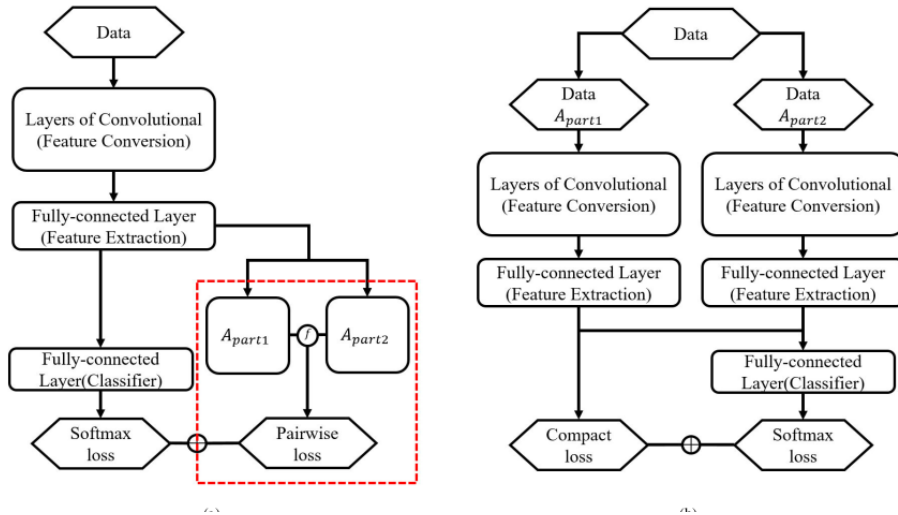
A função de custo Gaussiana pareada (*Pairwise Gaussian Loss - PGL*) ([Qin et al., 2020](#)) é baseada em princípios semelhantes do SCNet, no entanto, os dados são processados em uma rede de canal único com características divididas em dois grupos ao

final das camadas convolucionais. A Figura 3.4 mostra a diferença entre a arquitetura das redes utilizadas para a função de custo SCNet e PGL. A função de custo PGL usa uma função Gaussiana para calcular o contraste entre características e a função final é uma combinação da função de custo PGL e a *softmax loss*. A Equação 3.15 mostra a formulação da PGL,

$$\mathcal{L}_{PGL} = \mathcal{L}_{softmax} + \frac{4}{n^2} \sum_{i=1}^N \sum_{j=i+1}^N \left[\beta d_{ij}^2 + (y_{ij} - 1) \log(e^{\beta d_{ij}^2} - 1) \right] \quad (3.15)$$

onde y_{ij} é 1 se as amostras são da mesma classe e 0 caso contrário, d_{ij} é a distância Euclidiana entre dois vetores de características e β simplifica os parâmetros de escala da distribuição Gaussiana.

Figura 3.4: Diferença do funcionamento das redes SCNet e PGL, fonte: (Qin et al., 2020)



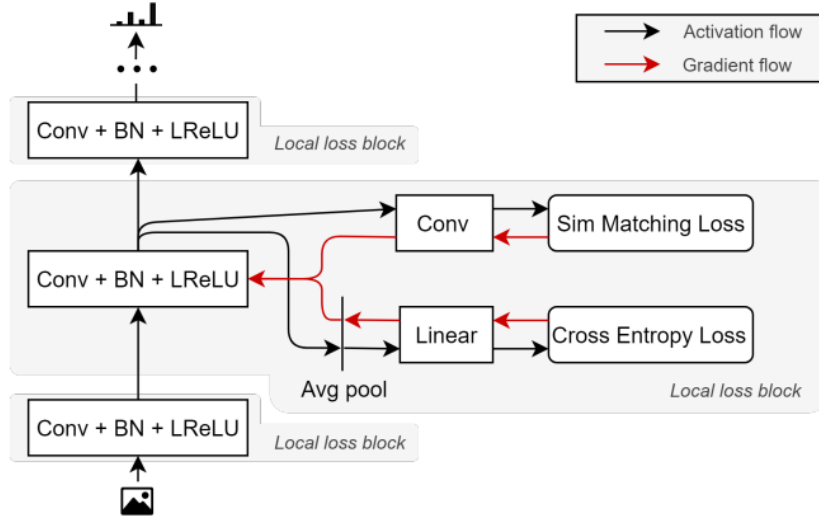
Uma função de custo com sinais locais de erro (*Predsim loss*) foi proposta por Nøkland & Eidnes (2019) e propõe utilizar o cálculo do erro em cada bloco da CNN, como mostra a Figura 3.5. A função de custo combina a função de custo *softmax loss* (L_{pred}) com uma função de custo de correspondência de similaridade (L_{sim}) que mede a distância $L2$ entre matrizes, onde os elementos contêm a similaridade de pares em um lote. A formulação de L_{sim} é dada pela Equação 3.16,

$$L_{sim} = \|S(\text{NeuralNet}(H)) - S(Y)\|_F^2 \quad (3.16)$$

onde H são as saídas da camada escondida, Y é uma matriz de rótulos (*one-hot encoded*), $S(X)$ é uma matriz de correlação de um lote X . Os elementos de $S(X)$ são dados pela Equação 3.17:

$$s_{ij} = \frac{\tilde{\mathbf{x}}_i^T \tilde{\mathbf{x}}_j}{\|\tilde{\mathbf{x}}_i\|_2 \|\tilde{\mathbf{x}}_j\|_2} \quad (3.17)$$

Figura 3.5: Esquema de funcionamento da função de custo com sinais locais de erro, fonte: (Nøkland & Eidnes, 2019)



onde o elemento \tilde{x}_i denota um vetor médio centrado x_i . A função de custo final é dada por,

$$\mathcal{L}_{predsim} = (1 - \beta)L_{pred} + \beta L_{sim} \quad (3.18)$$

Uma função de custo baseada na ideia de propriedade de localização foi desenvolvida por Liang *et al.* (2020). A função *LP-loss* trabalha com duas direções nas características extraídas, a direção principal (PE) e a direção secundária. A direção principal incorpora as funções de custo que realizará separação das características no espaço de projeção, os autores utilizaram *softmax loss*, *l-softmax* e *am-softmax* para esta direção. Para a direção secundária foi proposta a função de custo, chamada *S-OFP*, para realizar a compactação intraclasse das características. Para calcular a projeção secundária, é calculada a projeção das características em duas bases ortonormais definidas de acordo com a ortogonalização de Gram-Schmidt. Para evitar sobreajuste, em cada lote são escolhidas as k maiores características de acordo com a função de custo *S-OFP*. Depois de se obter 2 bases ortonormais, a projeção da representação das características Z_i em *S-OFP* é obtida. Se *softmax loss* for utilizada para a LP_{PE} a função de custo *LP loss* fica como descrito na Equação 3.19,

$$\mathcal{L}_{LP} = \alpha_1 \left(-\frac{1}{N} \sum_{i=1}^N y_i \log \hat{y}_i \right) + \alpha_2 \left(\frac{1}{2k} \sum_{i=1}^N G_i \|Z_i - P_{Z_i}\|^2 \right) \quad (3.19)$$

onde $\hat{y}_i = \sigma Z_i$, G é 1 se $i \in \eta_k$ e 0 caso contrário, η_k indica o conjunto do índice dos k maiores valores da função *S-OFP* no lote.

Uma CNN treinada com múltiplas funções de custo foi proposta por Xu *et al.* (2016), três funções foram utilizadas para realizar o treinamento: *softmax loss*, função de custo de posição pareada e a função *LambdaRank*, estas funções projetam espaços

de separação distintos. A função de custo de posição pareada, se inspira na *contrastive loss* e minimiza o erro de classificação considerando pares de amostras. A ideia é categorizar todas as imagens para que rótulos positivos (mesma classe, c^+) sempre tenham pontuações de predição mais altas do que rótulos negativos (classes diferentes, c^-). A formulação da função de custo de posição pareada é mostrada na Equação 3.20,

$$\mathcal{L}_{pairwise} = \frac{1}{N} \sum_{i=1}^N \sum_{m=1}^{c_i^+} \sum_{n=1}^{c_i^-} \max(0, 1 - q_m(f_i) + q_n(f_i)) \quad (3.20)$$

onde $q_m(f_i)$ é a probabilidade de f_i pertence a mesma classe e $q_n(f_i)$ de pertencer a classes diferentes.

A função de custo *LambdaRank* otimiza diretamente as k maiores acurácias de classificação, onde obtém os gradientes desejados diretamente, ao invés de derivá-los. As duas saídas $q_m(f_i)$ e $q_n(f_i)$ são mapeadas para a probabilidade aprendida sendo que o m -ésimo rótulo é categorizado maior que n -ésimo rótulo para uma função de ativação sigmoideal. A função descrita na Equação 3.21,

$$\mathcal{L}_{mn} = \log(1 + \exp(-\gamma(q_m(X_i) - q_n(X_i)))) \quad (3.21)$$

onde γ está relacionado a escala da função. Esta função de custo com múltiplas funções realiza o treinamento isolado com cada função de custo e realiza uma combinação por média dos classificadores na fase de predição da rede.

A função de custo de âncora (Ryou *et al.*, 2019) reavalia dinamicamente a dificuldade de classificação, determinada pela divergência entre as previsões verdadeiras e falsas. A formulação geral da função de custo por âncora é definida pela Equação 3.22,

$$l_{anchor} = - \underbrace{\left(1 + \overbrace{q - q^*}^{\text{dificuldade}}\right)^\gamma}_{\text{modulador}} \underbrace{(1 - p) \log(1 - q)}_{\text{entropia cruzada}} \quad (3.22)$$

onde p e q denotam os rótulos e as probabilidades preditas, respectivamente, $\gamma \geq 0$ é um hiperparâmetro que controla a faixa dinâmica da função de custo. A função de custo tem duas partes: modulador e entropia cruzada, onde o modulador é uma função monotônica crescente que considera a dificuldade de predição. Considerando ($q < q^*$), um caso com dificuldade de previsão fácil, a função de custo exclui amostras menos informativas ao atualizar o modelo. Em um caso moderado ($q = q^*$) a função de custo é equivalente a entropia cruzada, uma vez que o modulador torna-se 1. Por fim, em um caso difícil ($q > q^*$) a função de custo penaliza mais do que a entropia cruzada. Os autores utilizaram a função ativação sigmóide que trata a probabilidade de saída de cada classe como uma variável independente.

3.4 Resumo das funções de custo

Os trabalhos de novas funções de custo podem ser classificados em 3 categorias gerais: a) independentes da *softmax loss*, b) refinamentos da *softmax loss* e c) combinação da *softmax loss* com outras funções de custo.

- Os trabalhos independentes da *softmax loss* são: que utilizam a ideia de distância entre amostras.
 - *Contrastive loss* (Hadsell *et al.*, 2006): função de custo baseada no contraste entre duas amostras.
 - *Triplet loss* (Schroff *et al.*, 2015): utiliza três amostras para o contraste, sendo duas da mesma classe e a terceira de classe diferente.
- Os refinamentos da *softmax loss*, melhoram a projeção das características no espaço de separação forçando uma margem de separação ou modificando a distribuição das características neste espaço.
 - *L2-softmax* (Ranjan *et al.*, 2017): modifica a *softmax loss* para normalizar os pesos com a norma L2.
 - *L-softmax* (Liu *et al.*, 2016): insere uma margem angular na *softmax loss*.
 - *AM-softmax* (Wang *et al.*, 2018a): insere uma margem aditiva na *softmax loss*.
 - *SM-softmax* (Liang *et al.*, 2017): insere uma margem suave na *softmax loss*.
 - *G-softmax* (Luo *et al.*, 2020): reescreve a *softmax loss* para assumir que as características seguem a distribuição Gaussiana.
- *softmax loss* somada com termos que visam inserir um termo de regularização na função trazendo maior compactação intraclasse.
 - *Center loss* (Wen *et al.*, 2016): combina a *softmax loss* com distância de centros, onde é preciso atualizar a cada iteração um centro por classe.
 - *PEDCC loss* (Zhu *et al.*, 2020): realiza um refinamento da *center loss* onde os centros são pré-definidos e não é preciso atualizá-los durante o treinamento.
 - *Center loss* restrita (Shi *et al.*, 2021b): insere restrições de magnitude na *center loss* e propõe um método de atualização dos centros de forma analítica.
 - *SCnet* (Zhou *et al.*, 2019): combina a *softmax loss* com termo de compactação obtido a partir distâncias entre pares de características obtidas a partir de uma rede siamesa.

- PGL (Qin *et al.*, 2020): combina a *softmax loss* com um termo que utiliza distâncias entre pares de características obtidas após as camadas convolucionais da CNN.
- *Local loss* Nøkland & Eidnes (2019): realiza o cálculo da função de custo a nível de bloco convolucional e realiza combinação com a *softmax loss*.
- *LP loss* (Liang *et al.*, 2020): utiliza a combinação de dois classificadores, um com objetivo de obter a separação das classes (*softmax loss*) chamado de direção principal das características e um termo que tem objetivo trazer generalização que é chamado de direção secundária.
- *Anchor loss* (Ryou *et al.*, 2019): insere um termo de ponderação na função de custo de entropia cruzada, substitui a função de ativação *softmax* por sigmoideal.
- *EM-softmax* (Wang *et al.*, 2018b): utiliza a *softmax loss* junto com outros classificadores fracos obtidos sob critério de independência. Objetivo dos classificadores fracos é diversidade e generalização na solução final, que é obtida por média dos resultados dos classificadores.
- GCPL (Yang *et al.*, 2018): função de custo baseada em protótipos, combina a *softmax loss* com um termo para estressar a compactação intraclasse das características.
- *Multiloss* (Xu *et al.*, 2016): combina várias funções de custo para o classificador final, funde-se as probabilidades de diferentes funções de custo com a técnica de *average pooling*.

Capítulo 4

LDMAT

A função de custo LDMAT é calculada considerando um conjunto de características, $\{f_i\}_{i=1}^N$, extraídas por uma CNN a partir de um lote de N imagens, onde $f_i \in R^n$ é um vetor n -dimensional de características e $N \geq 2$. A matriz de distâncias $M_{N \times N}$, obtida a partir das características, contém os elementos m_{ij} que representam distâncias pareadas entre os vetores de características f_i e f_j . O uso de matrizes de distâncias permite analisar as distâncias em um lote de N amostras, diferente da *contrastive loss* e da *triplet loss*, que possuem limitação de duas ou três amostras, respectivamente. Neste trabalho a distância Euclidiana foi utilizada para construir uma matriz de distâncias como mostra na Equação 4.1,

$$m_{ij} = \sum_{k=1}^d (f_{ik} - f_{jk})^2 \quad (4.1)$$

onde d indica o número de características de cada amostras. Escolheu-se distância Euclidiana porque proporciona uma melhor compactação intraclasse, ao contrário de distância de representação angular, como a usada na *softmax loss* que realiza uma boa separação interclasse mas a compactação intraclasse é deficiente e dificulta a generalização do modelo. Uma segunda matriz de distâncias $D_{N \times N}$ é construída de modo similar, porém, ao invés de características, os rótulos das classes y (codificado pelo esquema *one hot*) são utilizados, como apresentado na Equação 4.2,

$$d_{ij} = \rho \left[\sum_{k=1}^d (y_{ik} - y_{jk})^2 + \xi S_{ij} \right] \quad (4.2)$$

onde S representa um termo de regularização diferenciável que permite um certo grau de relaxamento para a solução. Para esta perturbação foi utilizada uma onda senóide na matriz percorrendo cada linha onde $S_{i,j} = c \sin(i * N + j)$. Escolheu-se uma função senóide por ser uma função diferenciável o que facilita a implementação da função de custo, entretanto outras funções podem ser utilizadas. Esta perturbação produz um efeito serrilhado e torna a matriz de rótulos mais próxima da matriz de distâncias

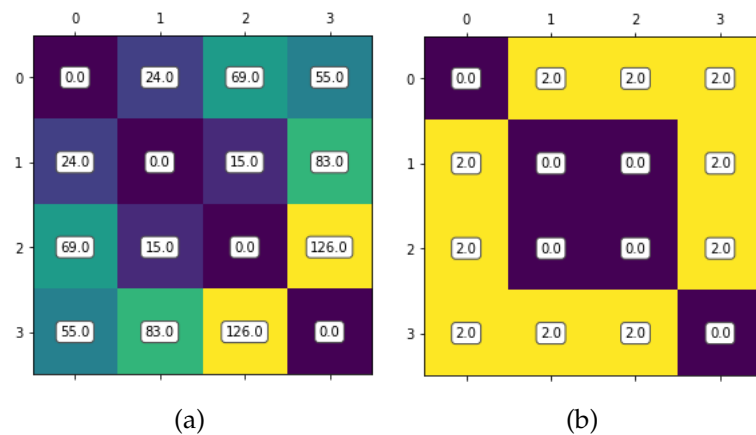


Figura 4.1: Exemplo de matrizes de distâncias, a) obtida pelas características e b) obtidas pelos rótulos.

geradas pelas características extraídas, como exibido na Figura 4.3. O termo c é um número aleatório limitado a um intervalo predefinido, amostrado, em cada execução da função de custo, de uma distribuição uniforme ($c \sim U(0, \xi)$) no intervalo $0 \leq c \leq \xi$. O termo de regularização permite o controle de generalização, de forma semelhante a técnica de suavização de rótulo usada em *Inception-V2* (Szegedy et al., 2016). ρ é a distância alvo entre as classes, $\rho > 0$, que permite criar um espaço maior de projeção das característica facilitando a discriminação das características em classes.

A função de custo LDMAT é construída a partir das matrizes M e D , e consiste em uma variação do RMSE, como mostrado na Equação 4.3. Utilizou-se a raiz quadrada da no cálculo da função de custo para se ter uma faixa de valores mais baixa dos valores de erro. Adicionalmente, uma constante foi utilizada para reescalar o valor final da função de custo, esta foi usada para evitar a ocorrência de estouro numérico devido a valores de erro muito altos no início do treinamento, este valor é multiplicado ao valor final da função de custo, por se tratar de um artifício computacional este não é definido da formulação da LDMAT.

$$\mathcal{L}_{DMAT}(f, y) = \left[\sum_{i=2}^N \sum_{j=1}^{i-1} (m_{ij} - d_{ij})^2 \right]^{1/2} \quad (4.3)$$

Um conjunto com 5 características e seus correspondentes rótulos são usados para apresentar um exemplo de construção das matrizes M e D . Considere as características $f = \{[1, 0, 0, 2, 4], [0, 2, 1, 5, 1], [0, 4, 0, 8, 0], [8, 1, 0, 1, 2]\}$ e seus respectivos rótulos: $y = \{[1, 0, 0], [0, 0, 1], [0, 0, 1], [0, 1, 0]\}$. As matrizes M e D são calculadas e mostradas nas Figuras 4.1(a) e 4.1(b). Para este exemplo, considerou-se $\xi = 0$ e $\rho = 1$. As amostras foram ordenadas de acordo com os seus respectivos rótulos para uma observação do bloco diagonal da matriz de distâncias. A função de custo LDMAT, entretanto, não requer que as amostras sejam ordenadas para realizar o treinamento. Observe

que a matriz de distâncias é simétrica e para os cálculos pode-se utilizar somente a matriz triangular superior ou inferior, a diagonal principal será sempre zero e pode ser ignorada nas operações entre matrizes de distâncias.

Outro exemplo para representar a funcionamento da função de custo LDMAT é apresentado na Figura 4.2, onde características de 3 classes são exibidas em um espaço bidimensional. Observe que o parâmetro ρ controla a distância entre as classes e ζ , o nível de espalhamento intraclasses. Utilizou-se os mesmos valores para ρ e ζ em toda a matriz de distância, não fazendo distinção por classe. Quando ζ é próximo de zero as características tendem a ficar muito próximas no espaço de representação e pode levar a um sobreajuste no modelo. No capítulo 5 é demonstrado como o efeito do parâmetro ζ influencia no espaço de projeção e conseqüentemente da taxa de acurácia.

Figura 4.2: Exemplo de representação de características de 3 classes em um espaço bidimensional.

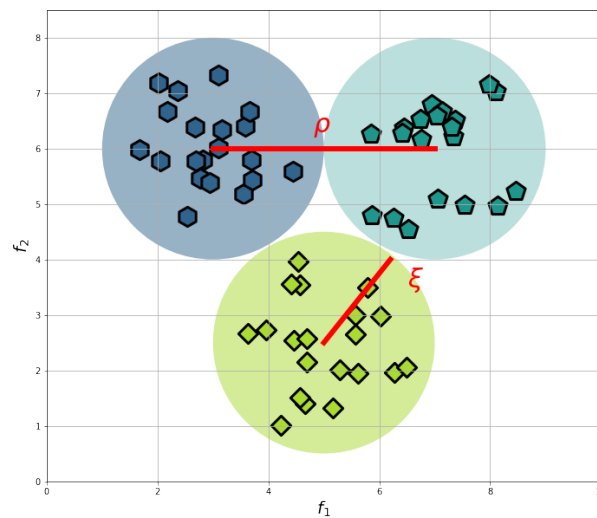
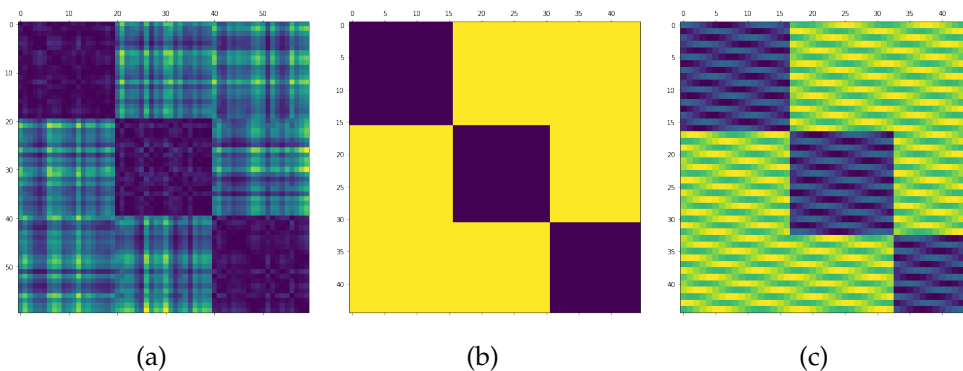


Figura 4.3: Matrizes de distâncias obtidas pelo exemplo da Figura 4.2, onde a) é matriz de distância das características, b) a matriz de distâncias dos rótulos e c) a matriz de distâncias dos rótulos com um nível de perturbação.



Na Figura 4.3 são apresentadas 3 matrizes de distâncias, a primeira é gerada a partir das características da Figura 4.2, observe que existe um nível serrilhamento na matriz da Figura 4.3(a) porque existem amostras que apesar classes diferentes podem ter uma proximidade maior entre elas do que amostra da mesma classe. Na Figura 4.3(b) é exibida uma matriz de distâncias obtidas a partir dos rótulos das classes, observe que a proximidade de amostras da mesma classe é sempre zero e a distância entre amostras de outras classes é sempre o mesmo valor, o que demonstra que esta matriz não representa fielmente um problema real. Para contornar este problema na Figura 4.3(c) é apresentada uma matriz de distâncias de rótulos que possui uma perturbação que insere uma variabilidade nas distâncias intraclasse e interclasse para se assemelhar um pouco mais com uma matriz de distâncias de características (Figura 4.3(a)). No processo de treinamento utilizando LDMAT o objetivo é aproximar a matriz de distâncias de características da matriz de distâncias de rótulos.

Já na Figura 4.4 são apresentadas duas representações bidimensionais de características obtidas a partir do treinamento no conjunto de dados CIFAR10. A primeira imagem (Figura 4.4(a)) representa o treinamento obtido com a função de custo *softmax loss*, onde pode-se observar que a separação das classes não é muito definida, principalmente nas amostras projetadas próximas ao centro da figura. As características utilizadas para produzir a figura foram obtidas na camada anterior a camada de saída da CNN treinada com a *softmax loss*. Já na Figura 4.4(b) as características são obtidas por uma rede treinada com a função de custo LDMAT observa-se que existe uma melhor separação interclasse e compactação intraclasse.

Os diagramas apresentados na Figura 4.5 ilustram a diferença entre o treinamento de uma CNN padrão que utiliza *softmax loss* e o treinamento de uma rede que utiliza a função de custo LDMAT. A abordagem convencional utiliza um treinamento fim-a-fim da rede com uma função de custo controlando o ajuste de todos os parâmetros. A proposta de treinamento em duas etapas utiliza a função de custo LDMAT para o treinamento somente da etapa de extração de características, na segunda etapa pode-se utilizar qualquer classificador, desta forma pode-se aplicar penalidades diferentes em cada etapa do treinamento.

Figura 4.4: Visualização das características obtidas em dois treinamentos distintos: a) obtida a *Softmax Loss* e b) obtida com o treinamento da LDMAT. A dimensionalidade das características foi reduzida utilizando a redução de dimensionalidade t-SNE (Belkina *et al.*, 2019). Foi utilizada o conjunto de dados CIFAR10 e o treinamento foi feito com a arquitetura LiuNet-B.

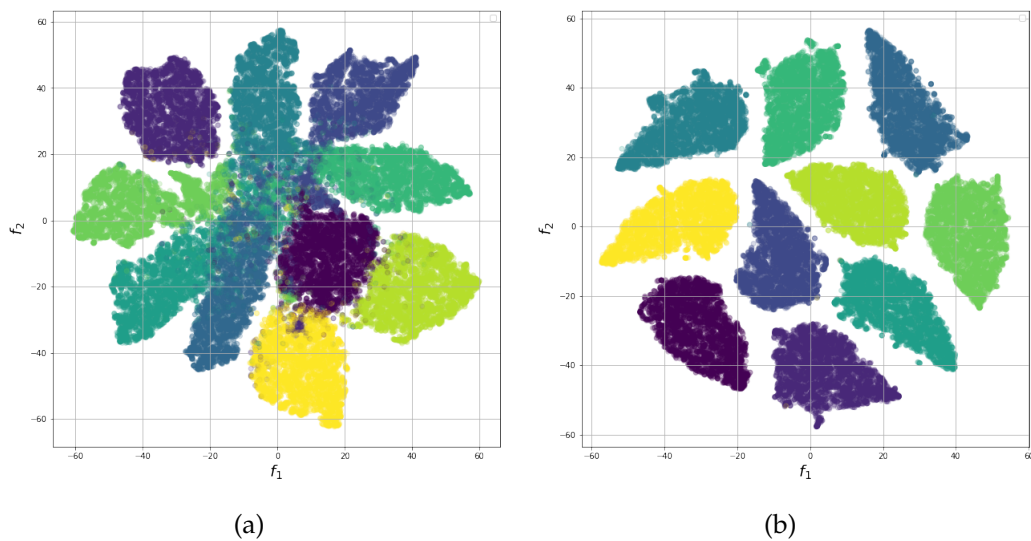
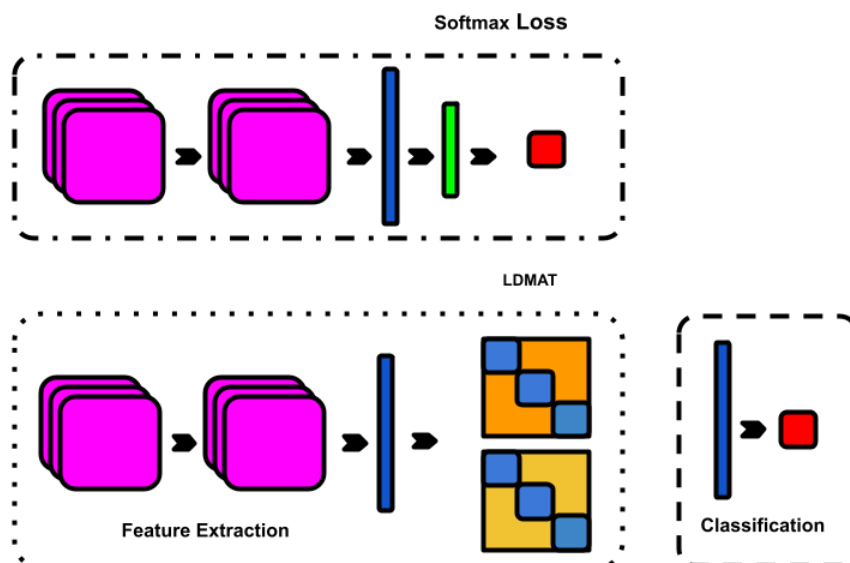


Figura 4.5: Diagrama ilustrando a diferença entre o modo de treinamento de uma CNN padrão e um treinamento em duas etapas utilizando a função de custo LDMAT. Os blocos em rosa indicam as camadas convolucionais, o retângulo em azul indica as características extraídas pelos blocos convolucionais, o retângulo em verde indica as camadas totalmente conectadas e o quadrado em vermelho indica a função de custo *softmax loss*. A segunda figura indica que o treinamento utilizando a LDMAT é feito em duas etapas distintas indicado pela separação pelas linhas tracejadas. As duas matrizes em laranja e amarelo indicam as matrizes de distâncias de características e de rótulos, utilizadas pelo função de custo LDMAT.



4.1 Otimização

O processo mais comum para o treinamento de uma rede neural é realizar um processo iterativo de otimização onde os pesos do modelo são ajustados de acordo com a aproximação do gradiente do erro produzido pela função de custo. O cálculo do gradiente para ser utilizado na etapa de retropropagação do erro na função de custo LDMAT é mostrado nas Equações 4.4 até 4.8. Primeiramente, a diferenciação é feita em relação a f na Equação 4.3.

$$\frac{\partial L}{\partial f} = \frac{\partial}{\partial f} \left[\sum_{i=2}^N \sum_{j=1}^{i-1} (m_{ij} - d_{ij})^2 \right]^{1/2} \quad (4.4)$$

Com a aplicação da regra da cadeia e derivando em relação a f_{ij} obtém-se a Equação 4.5.

$$\frac{\partial L}{\partial f} = \left(\frac{1}{2L} \right) \frac{\partial}{\partial f_{ij}} (m_{ij} - d_{ij})^2 \quad (4.5)$$

Continuando o processo com respeito a m_{ij} e d_{ij} , chega-se a Equação 4.6.

$$\frac{\partial L}{\partial f} = \left(\frac{1}{2L} \right) 2 (m_{ij} - d_{ij}) \frac{\partial}{\partial f_{ij}} (m_{ij} - d_{ij}) \quad (4.6)$$

Substituindo m_{ij} com f a partir da Equação 4.1, onde n é o número de características, estende-se o cálculo para a Equação 4.7.

$$\frac{\partial L}{\partial f} = \left(\frac{1}{L} \right) (m_{ij} - d_{ij}) \frac{\partial}{\partial f_{ij}} \left(\sum_{k=1}^n (f_{ik} - f_{jk})^2 - d_{ij} \right) \quad (4.7)$$

A equação final do cálculo do gradiente é dada pela Equação 4.8.

$$\frac{\partial L}{\partial f_{ij}} = \left(\frac{2}{L} \right) \sum_{k=1}^n (f_{ik} - f_{jk}) (m_{ij} - d_{ij}) \quad (4.8)$$

Observe que é necessário calcular a distância entre as amostras f_i e f_j e para fazer isto é necessário computar um somatório. No capítulo 5, foi feita uma análise do tempo de treinamento de uma CNN com a função de custo LDMAT em comparação com redes neurais convolucionais treinadas fim-a-fim com a função de custo *softmax loss*. Percebe-se com esta análise que o uso de matrizes de distâncias no treinamento de uma CNN não eleva o custo computacional para treinar o modelo de uma CNN.

4.2 Algoritmo

Pode-se condensar todos os passos para o treinamento de uma CNN com a função de custo LDMAT em uma sequência de passos como descrito no Algoritmo 1. Deve-se

utilizar uma rede convolucional sem as camadas densas, após os filtros convolucionais é feito uma vetorização das características para uma representação unidimensional.

Algorithm 1 Treinamento de uma CNN com a função de custo LDMAT

```

1: iniciar os pesos do modelo
2: while não convergir, faça: do
3:   for cada lote de  $N$  imagens,  $\tau_u = \{x_i; y_i\}_{i=1}^N$  do
4:     realizar a alimentação da rede,  $f_i = \text{feedforward}(x_i)$ 
5:     for cada imagem  $i$  no lote do
6:       for cada  $j$  imagem no lote do
7:         Calcular  $m_{ij} = \sum_{k=1}^n (f_{ik} - f_{jk})^2$ 
8:         Calcular  $S_{i,j} = \sin(i * N + j) * c$ ,  $c \sim U(0, \xi)$ 
9:         Calcular  $d_{ij} = \rho \left[ \sum_{k=1}^n (y_{ik} - y_{jk})^2 + S_{ij} \right]$ 
10:        end for
11:       end for
12:       Computar LDMAT:  $L(f, y) = \left[ \sum_{i=2}^N \sum_{j=1}^{i-1} (m_{ij} - d_{ij})^2 \right]^{1/2}$ 
13:       Retropropagar o erro
14:     end for
15: end while

```

4.3 Estratégia de combinação (*ensemble*)

O uso do parâmetro de perturbação na matriz de distâncias de rótulos tem objetivo de produzir uma matriz de distâncias que se aproxima de um problema real. Os diversos níveis de perturbação podem cobrir maior o espaço de representação de características produzindo uma melhor generalização do problema. Pode-se combinar em uma única solução diversos conjuntos de características geradas a partir de treinamentos com diferentes níveis de perturbação. Quando não se aplica nenhuma perturbação ($\xi = 0$), obtém-se características orientadas em gerar um superfície de separação entre as classes, porém com $\xi > 0$, tem-se uma maior relaxamento na solução o que introduz uma maior variabilidade ao problema. Considerando-se que exista K conjuntos de características, cada um obtido com um nível de regularização, pode-se obter o conjunto de características combinadas utilizando uma média das características correspondentes como a Equação 4.9 mostra.

$$f_{ij_E} = \frac{1}{K} \sum_{i=1}^K f_{ij_k} \quad (4.9)$$

A técnica de utilizar combinação de diversas soluções foi utilizada no trabalho de [Xu et al. \(2016\)](#) e [Wang et al. \(2018b\)](#).

4.4 Conclusão do Capítulo

Neste capítulo foi apresentada a formulação e análise da função de custo LDMAT. Optou-se pelo treinamento em duas etapas para se treinar diretamente as características, separando-as da etapa de classificação. Na função de custo *contrastive loss* (Hadsell *et al.*, 2006) é feito o treinamento diretamente das características, assim como no trabalho de Yang *et al.* (2018). O treinamento com o uso das características separadas permite utilizar qualquer classificador com as características extraídas e permite de forma mais fácil realizar o uso de combinação de soluções. O uso da análise das distâncias entre amostras com o uso de matrizes de distâncias permite o treinamento de dados desbalanceados e sem a necessidade de ordenação das amostras. O uso de perturbação na matriz de distâncias de rótulos produz um efeito de generalização no problema evitando sobreajuste dos pesos do modelo.

A função de custo LDMAT não utiliza combinação com a função de custo *softmax loss*, optou-se por fazer o treinamento diretamente nas características com o uso da distância Euclidiana para calcular as distância entre as amostras por esta apresentar uma maior compactação intraclasse. A função de custo *softmax loss* induz uma separação angular no espaço de características e parte dos trabalhos da literatura buscam mitigar a baixa compactação intraclasse da *softmax loss* a combinando com termos que utilizam a distância euclidiana entre amostras.

Capítulo 5

Experimentos

Experimentos foram realizados para avaliar o desempenho da função de custo LDMAT e realizar comparação com os trabalhos publicados na literatura. Os conjuntos de dados utilizados foram selecionados baseados nos utilizados em trabalhos publicados da área de classificação de imagens com redes convolucionais, o mesmo ocorre com a arquitetura da CNN utilizada para verificar o desempenho da função de custo LDMAT.

5.1 Conjuntos de Dados

Buscou-se conjuntos de dados frequentemente utilizados em pesquisas de aprendizado profundo e selecionou-se 6 conjuntos de dados de imagens utilizados em tarefas de classificação com redes neurais convolucionais. Na Tabela 5.1 são apresentados os detalhes dos conjuntos de imagens selecionados e na Figura 5.1 existe a seleção de algumas imagens de cada conjunto de dados. Uma particularidade destes conjuntos de dados é que já existe uma divisão entre os conjuntos de treinamento e o conjunto de testes, isto torna mais fácil a comparação de desempenho entre diversos modelos de aprendizagem.

O conjunto de dados MNIST trata a tarefa de classificação de dígitos manuscritos (10 classes), esta base de dados foi utilizada inicialmente por [LeCun et al. \(1989\)](#) e está presente em diversos trabalhos. Atualmente os resultados alcançados por CNNs na base MNIST estão saturados e próximos de 100%. Já o conjunto de dados *Fashion MNIST* ([Xiao et al., 2017](#)) é uma variação da MNIST que utiliza imagens em escala de cinza de peças de vestuário. A base de dados *Extended MNIST*¹, inclui além dos dígitos, caracteres alfanuméricos que estão distribuídos em 47 classes, para este conjunto de dados foi utilizada a base de dados balanceada.

Já no conjunto de dados SVHN (*Street View House Number*)² existem imagens co-

¹<https://www.nist.gov/itl/products-and-services/emnist-dataset>

²<http://ufldl.stanford.edu/housenumbers/>

loridas de numeração de residencias obtidas pelo *Google Street View*. Nesta base não foi utilizada a base de dados extras que possui 531131 amostras adicionais de treinamento. Os conjuntos de dados CIFAR₁₀ e CIFAR₁₀₀ possuem 10 e 100 classes respectivamente, e são compostos por imagens naturais como animais e automóveis, as imagens são coloridas e foram utilizadas inicialmente no trabalho de [Krizhevsky \(2009\)](#).

Tabela 5.1: Descrição dos conjuntos de imagens utilizados no experimentos.

Nome	Domínio	#Classes	Dimensão	#Trein.	#Teste
MNIST (LeCun et al., 2010)	números	10	28x28x1	60000	10000
FMNIST (Xiao et al., 2017)	vestuário	10	28x28x1	60000	10000
EMNIST (Cohen et al., 2017)	letras	47	28x28x1	112799	18799
SVHN (Netzer et al., 2011)	números	10	32x32x3	73257	26032
CIFAR ₁₀ (Krizhevsky, 2009)	naturais	10	32x32x3	50000	10000
CIFAR ₁₀₀ (Krizhevsky, 2009)	naturais	100	32x32x3	50000	10000

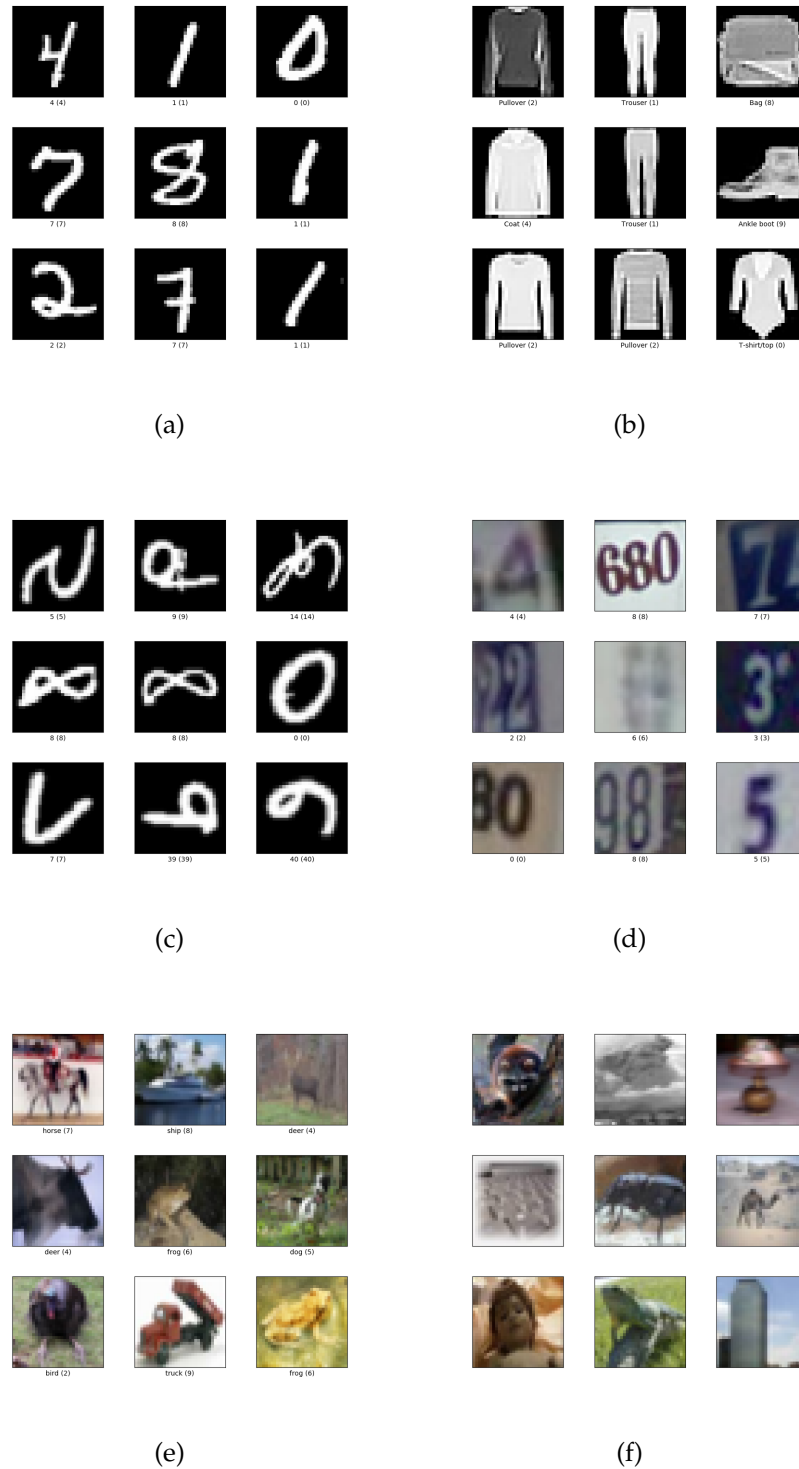


Figura 5.1: Imagens para ilustrar os conjuntos de dados utilizados nos experimentos: MNIST, *Fashion MNIST*, *Extended MNIST*, *Street View House Number*, CIFAR₁₀ e CIFAR₁₀₀.

5.2 Arquiteturas

Para treinar as bases de dados selecionadas foram utilizadas arquiteturas de uma rede neural convolucional inspirada na rede VGG (Simonyan & Zisserman, 2015), estas arquiteturas são descritas no trabalho de Liu *et al.* (2016) e são mostrada nas Tabelas 5.2 e 5.3, por conveniência estas arquiteturas são referidas como LiuNet. A arquitetura LiuNet-A foi utilizada com as bases de dados em escala de cinza (MNIST, FMNIST e EMNIST), a arquitetura LiuNet-B foi utilizada em bases de dados coloridas (CIFAR₁₀ e SVHN) e a arquitetura LiuNet-C é utilizada com o conjunto de dados CIFAR₁₀₀. Nas arquiteturas LiuNet existe o uso da técnica de normalização de lotes (Ioffe & Szegedy, 2015) seguida pela função de ativação ReLU, ambos utilizados depois de cada camada convolucional. Depois de cada bloco convolucional é utilizada uma camada de desligamento de conexões (*dropout*). Estas arquiteturas foram escolhidas porque elas foram utilizadas nos trabalhos de Liang *et al.* (2017), Wang *et al.* (2018b), Zhou *et al.* (2019), Qin *et al.* (2020) e Zhu *et al.* (2020), o que torna mais justa a comparação dos resultados obtidos com a função de custo LDMAT com estes resultados já publicados. Para os experimentos com a função LDMAT a camada totalmente conectada e a camada de saída foram removidas e realizou-se experimentos para verificar o desempenho da combinação das técnicas de *dropout* com o uso de perturbação na matriz de distâncias de rótulos (Tabela 5.5).

Tabela 5.2: Arquitetura utilizada nos experimentos. Proposta no trabalho de Liu *et al.* (2016), chamou-se esta arquitetura de LiuNet e esta foi utilizada como critério de comparação por diversos trabalhos publicados. Por exemplo [3x3,64],3 denota 3 camadas convolucionais com 64 filtros de tamanho 3x3

Camada	LiuNet-A	LiuNet-B
Conv1	Convolução [3x3,64] x 4	Convolução [3x3,64] x 5
Pool1	MaxPooling 2x2	MaxPooling 2x2
Drop1	Dropout1	Dropout1
Conv2	Convolução [3x3,64] x 3	Convolução [3x3,96] x 4
Pool2	MaxPooling 2x2	MaxPooling 2x2
Drop2	Dropout2	Dropout2
Conv3	Convolução [3x3,64] x 3	Convolução [3x3,128] x 4
Pool3	MaxPooling 2x2	MaxPooling 2x2
Drop3	Dropout3	Dropout3
Flatten	Flatten	Flatten
Totalmente conectadas	256	256
Full Connected	10	10

Tabela 5.3: Arquitetura utilizada nos experimentos, Proposta no trabalho de Liu *et al.* (2016) e utilizada com conjunto de dados CIFAR100.

Camada	LiuNet-C
Conv1	Convolução [3x3,96] x 5
Pool1	MaxPooling 2x2
Drop1	Dropout1
Conv2	Convolução [3x3,192] x 4
Pool2	MaxPooling 2x2
Drop2	Dropout2
Conv3	Convolução [3x3,384] x 4
Pool3	MaxPooling 2x2
Drop3	Dropout3
Flatten	Flatten
Full Connected	512
Full Connected	100

Verificou-se o custo computacional de se utilizar as arquiteturas LiuNet em relação ao número de operações de ponto flutuante e o tempo de execução, como mostrado na Tabela 5.4. Apesar do custo computacional para se obter as matrizes de distâncias em cada iteração do treinamento, a arquitetura que utiliza a função LDMAT (indicação s/FC na Tabela 5.4) possui um tempo de execução menor que as redes com a função de custo *softmax loss*. Existe também uma redução dos parâmetros por não utilizar as camadas totalmente conectadas. Porém, o método de treinamento que utiliza uma a função de custo LDMAT necessita de um outro modelo para realizar a classificação.

A classificação com a função de custo LDMAT consiste de dois estágios, extração das características da função de custo LDMAT e classificação com um classificador arbitrário. Por conveniência, uma rede *perceptron* com somente uma camada de saída é utilizada no segundo estágio porque a função de custo LDMAT projeta uma separação linear no espaço das características.

Tabela 5.4: Análise do número de parâmetros da rede LiuNet em relação ao número de operações de ponto flutuante por segundo (FLOPS) na escala de 10^9 e o tempo gasto para a execução de uma época treinamento (executado em uma GPU Tesla K80). Para treinar a rede LiuNet-A usou-se o conjunto de dados MNIST e LiuNet-B usou-se CIFAR10. A indicação s/FC significa que a arquitetura de rede utilizada não tem a camada totalmente conectada e a camada de saída.

Rede	Parâmetros	FLOPS	Tempo época
LiuNetA	485 834	0,229 G	58s 30ms
LiuNetB	1 539 466	0,534 G	80s 50ms
LiuNetC	9 351 908	1,881 G	144s 90ms
LiuNetA(s/FC)	335 552	0,229 G	54s 28ms
LiuNetB(s/FC)	1 012 352	0,533 G	76s 47ms
LiuNetC(s/FC)	6 154 368	1,874 G	134s 86ms

5.3 Detalhes de Implementação

A execução de um experimento em modelos de aprendizagem em profundidade é muito custoso computacionalmente e majoritariamente são conduzidos com o uso de unidades de processamento gráfico (*Graphics Processing Unit* - GPU) para realizar as operações de ajuste do modelo. Por causa do custo computacional dispensado para treinar um modelo de rede neural convolucional, não é comum encontrar trabalhos que realizam diversas execuções do mesmo modelo em um mesmo conjunto de dados. Nos conjuntos de dados frequentemente utilizados já existe uma divisão de conjunto de treinamento e conjunto de teste, isto permite uma comparação entre os modelos, porém o desempenho do modelo pode variar de acordo com os valores iniciais dos pesos e a ordem de apresentação das imagens à CNN. Considerando isto é comum fixar a semente utilizada para gerar os números pseudo-aleatórios, assim tem-se uma comparação justa com o modelo de referência, porque são gerados os mesmos pesos iniciais do modelo, a mesma divisão em lotes e demais valores gerados aleatoriamente durante o treinamento. Entretanto, pode-se executar diversas vezes o mesmo treinamento para fixar uma semente que apresenta bom resultado para o método desejado. Para os experimentos do presente trabalho foi escolhido fixar uma semente para a geração dos números pseudo-aleatório em cada conjunto de dados, todavia não realizou-se uma busca intensiva para escolher uma semente que apresenta bons resultados no modelo desejado. Utilizou-se a linguagem Python com a biblioteca Keras¹ para a escrita do código da função de custo e toda a parte operacional da execução dos experimentos. Uma unidade de processamento gráfico

¹<https://keras.io/>

(GPU Nvidia 1080) foi utilizada para realizar os experimentos e as tarefas de análise de dados e teste de códigos foram feitas na ferramenta *Google Colaboraty*¹.

Foi utilizado como otimizador dos parâmetros o método do gradiente descendente estocástico (*Stochastic Gradient Descent - SGD*). Um mini-lote de tamanho 16 foi usado, o que implica que a matriz de distâncias na função de custo LDMAT em tamanho 16x16. Os pesos são iniciados de acordo com o método proposto por He *et al.* (2015), a regularização de redução de valor do peso com $\lambda = 5 \cdot 10^{-4}$ (*weight decay*) é considerada em cada camada convolucional e os *pixels* da imagem de entrada são redimensionados para o intervalo $[0, 1]$, assim como utilizado no trabalho de Liu *et al.* (2016) e todos os subseqüente que utilizam a arquitetura LiuNet. A técnica de aumento de dados que foi utilizada consiste em fazer um espelhamento horizontal nas imagens de entrada de forma aleatória e a inserção de bordas nas imagens (com valor 0 e espessura de 4 *pixels*) seguido de um recorde aleatório do quadro de tamanho da imagem original (*random crop*).

Como parâmetro da função de custo LDMAT selecionou-se a distância média entre as classes com o valor $\rho = 1024$, para chegar neste valor foram feitos testes empíricos. Utilizou-se um artifício de reduzir a escala dos valores da função de custo a para uma melhor análise da convergência, para tal multiplicou-se por $\frac{1}{2^{15}}$. No início do treinamento, uma estratégia de aquecimento gradual que aumenta linearmente a taxa de aprendizagem foi adotada (P. Goyal & He, 2017). Após o aquecimento, de 5 épocas, outras 1000 épocas de treinamento são processadas, com queda gradativa da na taxa de aprendizagem (Loshchilov & Hutter, 2017), sendo que para cada época, a taxa de aprendizagem η é calculada de acordo com a Equação 5.1,

$$\eta_t = \frac{1}{2} \left[1 + \cos \left(\frac{t\pi}{T} \right) \right] \eta \quad (5.1)$$

onde T representa o número total de épocas e $\eta = 1 \cdot 10^{-2}$ foi escolhida para a taxa de aprendizado inicial. Foram realizados experimentos e o treinamento com mais de 1000 épocas não diminuiu o erro de treinamento nem aumentou a acurácia dos conjuntos de dados.

A definição do parâmetro ξ , assim como a estratégia de desligamento de pesos em cada camada é apresentada na Tabela 5.5, onde foi utilizado o conjunto de dados CIFAR10, com uso de aumento de dados, para treinar cada modelo.

Na Figura 5.2 são exibidas as curvas de convergências de cada uma combinação exibidas na Tabela 5.5, verificou-se que todas as combinações de parâmetros convergem e o que a combinação que atinge menor valor de erro da função de custo LDMAT é a combinação de desligamento de conexões (*dropout*) com 25% de desligamento após o primeiro e o segundo bloco convolucional e treinado em conjunto com

¹colab.research.google.com

Tabela 5.5: Escolha da combinação das técnicas de desligamento de conexões e perturbação na matriz de distâncias de rótulos. Foi utilizada o conjunto de dados CIFAR10+ e rede LuiNet-B. Drop₁, Drop₂ e Drop₃ representam o nível de desligamento de conexões no final das camadas convolucionais como exibido nas Tabelas 5.3 e 5.3. ξ representa o nível de perturbação na matriz de distâncias de rótulos da LDMAT. A coluna acc indica a acurácia obtida pela combinação de parâmetros de regularização na etapa de classificação.

regularização	drop ₁	drop ₂	drop ₃	ξ	acc
θ_1	0,0	0,0	0,0	0,0	91,58
θ_2	0,0	0,0	0,0	0,25	94,08
θ_3	0,25	0,0	0,0	0,0	92,22
θ_4	0,25	0,0	0,0	0,25	94,46
θ_5	0,25	0,25	0,0	0,0	94,46
θ_6	0,25	0,25	0,0	0,25	94,16
θ_7	0,25	0,25	0,25	0,0	91,35
θ_8	0,25	0,25	0,25	0,25	94,19

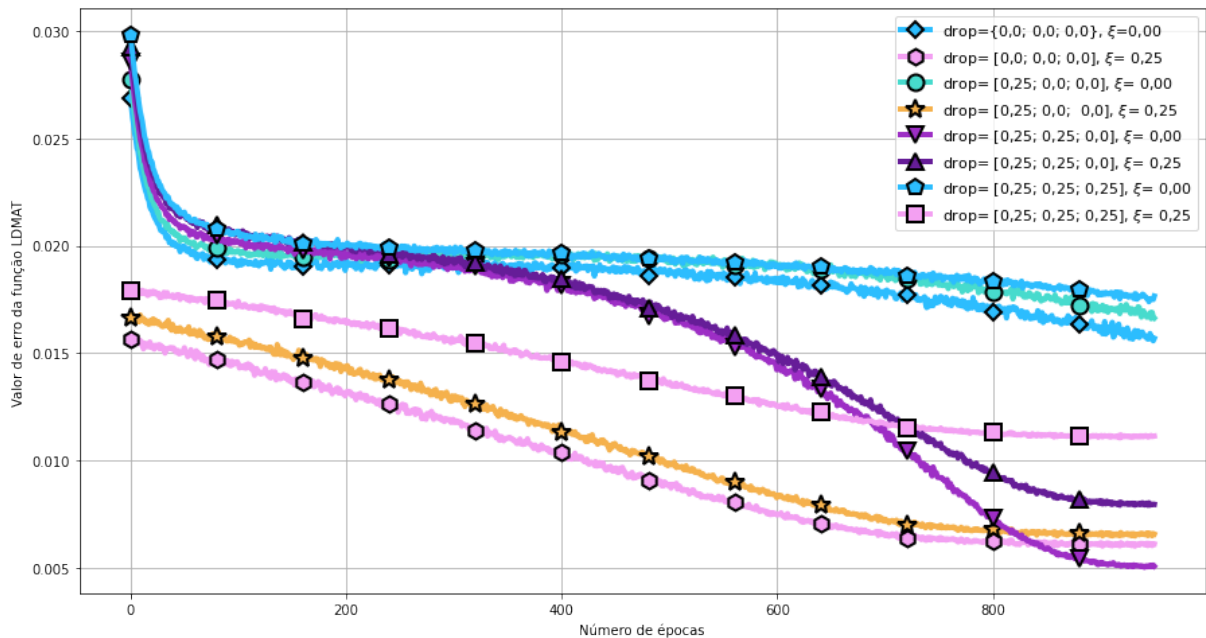
nenhuma perturbação na matriz de rótulos ($\xi = 0$). Para uma melhor visualização os valores de erro das 50 primeiras épocas foram omitidos porque apresentaram valores muito altos.

Observando a Tabela 5.5 verifica-se que o uso de perturbação na matriz de distância de rótulos (linha 2) melhora o resultado em relação de quando não se usa nenhuma regularização (linha 1).

Já na Figura 5.3 são apresentadas as taxas de acurácia a cada 100 épocas de treinamento, para gerar estes resultados foram salvos os pesos durante o treinamento, com a periodicidade de 100 épocas, posteriormente estes pesos foram carregados na rede, foi feita uma alimentação da CNN com a base treinamento e teste para a obtenção das características extraídas. Após isto, um classificador *perceptron* foi utilizado para obter o valor de acurácia. Observa-se que as combinações que apresentaram os melhores resultados tem taxas de acurácia muito próximas. Os piores resultados ficaram com as redes que foram treinadas a) sem nenhuma abordagem de regularização(drop=0,0; 0,0; 0,0, $\xi = 0,00$), b) com *dropout* somente no primeiro bloco convolucional e c) com *dropout* em todos os blocos convolucionais em sem perturbação na matriz de distâncias. Isto demonstra que a perturbação inserida isoladamente na matriz de rótulos durante o treinamento da rede com a função LDMAT produz efeito similar a diversas configurações de *dropout*.

Para o uso de um conjunto de soluções com a função de custo LDMAT, selecionou-

Figura 5.2: Curvas de convergência da função de custo LDMAT com variações do nível de perturbação na matriz de distância de rótulos e o uso do *dropout*. Foi utilizado o conjunto de dados CIFAR10 com aumento de dados e a rede foi treinada com a arquitetura LiuNet-B.



se as características obtidas pelas redes treinadas com as configurações de regularização que apresentaram melhores resultados. Os resultados são tabulados na Tabela 5.6, observa-se que existe uma melhora de acurácia com o uso do conjunto de classificadores em 1,18% na base de dados CIFAR10, com aumento da dados.

Figura 5.3: Acurácia obtida ao longo do treinamento de uma CNN com a função de custo LDMAT. Foi utilizado o conjunto de dados CIFAR10 com aumento de dados e a arquitetura LiuNet-B. Os pesos foram salvos a cada 100 épocas e posteriormente as características foram obtidas e utilizadas em um classificador (*perceptron* linear) para a obtenção da acurácia.

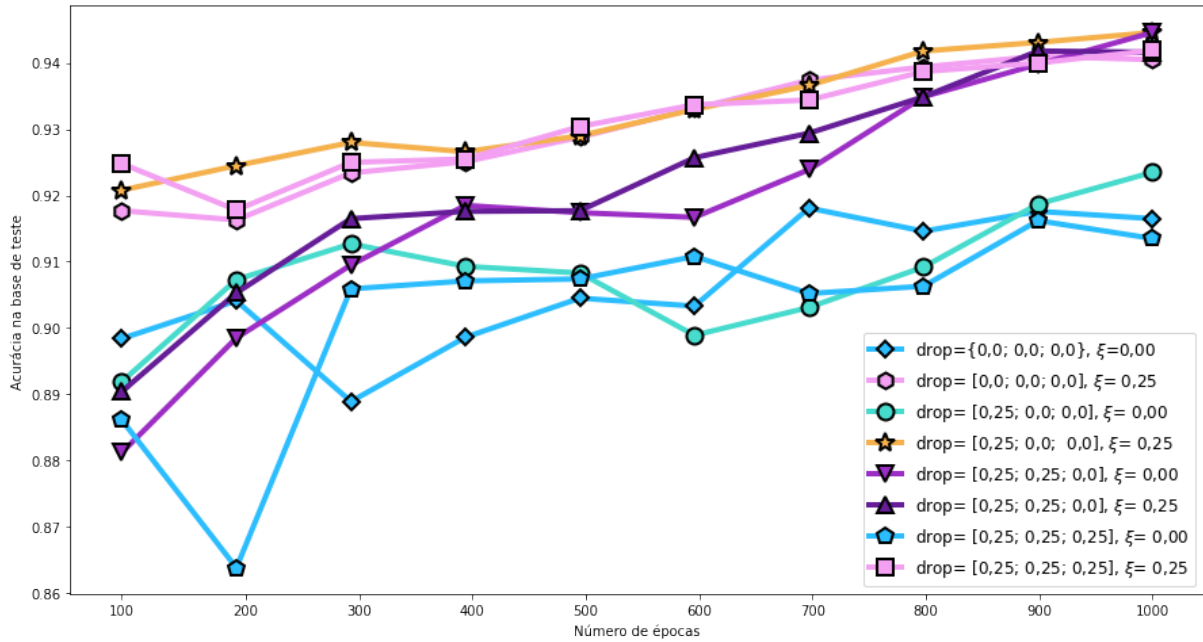


Tabela 5.6: Relação das características utilizadas para o conjunto de soluções. Nas primeiras linhas são apresentadas a taxa de acurácia individual de cada combinação de parâmetros e na última linha a acurácia com a fusão das características.

regularização	drop1	drop2	drop3	ζ	acurácia
θ_2	0,0	0,0	0,0	0,25	94,08
θ_4	0,25	0,0	0,0	0,25	94,46
θ_5	0,25	0,25	0,0	0,0	94,46
θ_6	0,25	0,25	0,0	0,25	94,16
θ_8	0,25	0,25	0,25	0,25	94,19
LDMAT-Conjunto					95,64

5.4 Acurácia

A qualidade das características extraídas são avaliadas de acordo com o desempenho em conjunto de dados na tarefa de classificação. As tabelas de resultados foram construídas a partir dos resultados obtidos pela função de custo LDMAT e os resultados publicados pelos trabalhos na literatura. Os conjuntos de dados mais utilizados na literatura são: MNIST, CIFAR₁₀ e CIFAR₁₀₀, entretanto com os conjuntos de dados EMNIST, FMNIST e SVHN existe pelo menos um trabalho que utilizou o conjunto de dados nos experimentos, como relacionado nas Tabelas 5.8, 5.9 e 5.12.

A Tabela 5.7 apresenta os resultados obtidos pela rede LiuNet-A no conjunto de dados MNIST, esta rede foi treinada com a função de custo LDMAT e os resultados são comparados com diversos trabalhos publicados que propuseram abordagens para funções de custo para CNNs. Para uma comparação justa, a tabela apresenta a arquitetura utilizada na publicação original. O resultado utilizado como base de comparação (*baseline*) foi a rede treinada com a função de custo *softmax loss*, este resultado foi obtido por execução própria de treinamento.

Tabela 5.7: Resultados publicados que utilizam o conjunto de dados MNIST e os resultados da função de custo LDMAT.

Método	Arquitetura	MNIST
Constrative (Hadsell <i>et al.</i> , 2006)	LeNet	98,80
Triplet (Schroff <i>et al.</i> , 2015)	Inception	99,06
Multiloss (Xu <i>et al.</i> , 2016)	NiN	99,58
L-Softmax (Liu <i>et al.</i> , 2016)	LiuNet-A	99,69
SM-Softmax (Liang <i>et al.</i> , 2017)	LiuNet-A	99,70
GCPL (Yang <i>et al.</i> , 2018)	ResNet32	99,33
EM-Softmax (Wang <i>et al.</i> , 2018b)	LiuNet-A	99,73
SCNet (Zhou <i>et al.</i> , 2019)	LiuNet-A	94,34
PGL (Qin <i>et al.</i> , 2020)	LiuNet-A	99,83
Local loss (Nøkland & Eidnes, 2019)	VGG-8B	99,74
Softmax Loss	LiuNet-A	99,62
LDMAT	LiuNet-A(s/FC)	99,72
LDMAT-Conjunto	LiuNet-A(s/FC)	99,74

No conjunto de dados MNIST existem 10.000 amostras no conjunto de teste, isto implica que a cada 0,01% na taxa de acurácia equivale a classificação de 1 imagem na fase de predição. Isto demonstra que as abordagens propostas apresentam resultados muito próximos na classificação da base MNIST. A melhora que LDMAT apresenta em relação a *softmax loss* é de 0,1%, porém a melhora do uso de LDMAT com fusão

das característica é de 0,02%. O resultado publicado por [Qin et al. \(2020\)](#) supera o resultado de LDMAT com fusão de dados, e a publicação de [Nøkland & Eidnes \(2019\)](#) apresenta resultado similar, os demais resultados publicados apresentam resultados inferiores.

Tabela 5.8: Resultados com conjunto de dados *Extended MNIST*.

Método	Arquitetura	EMNIST
PEDCC Zhu et al. (2020)	LiuNet-A	89,83
Softmax Loss	LiuNet-A	88,42
LDMAT	LiuNet-A(s/FC)	90,08
LDMAT-Conjunto	LiuNet-A(s/FC)	90,40

Na Tabela 5.8 são apresentados os resultados no conjunto de dados EMNIST. A execução do treinamento foi feita com rede LiuNet-A e os resultados da função de custo LDMAT são comparados com o resultado do método *PEDCC loss* ([Zhu et al., 2020](#)). LDMAT apresenta melhor acurácia em relação a *softmax loss* de 1,66% e a melhora do uso da fusão de soluções em relação a solução única com a LDMAT é de 0,32%. Já a melhora do uso de um conjunto de soluções em relação ao melhor resultado publicado, *PEDCC loss*, é de 0,57%.

Tabela 5.9: Resultados com conjunto de dados *Fashion MNIST*.

Método	Arquitetura	#parâmetros	FMNIST
Local Loss Nøkland & Eidnes (2019)	VGG-8B	7,3M	95,35
Softmax Loss	LiuNet-A	0,4M	94,34
LDMAT	LiuNet-A(s/FC)	0,3M	95,30
LDMAT-Conjunto	LiuNet-A(s/FC)	1,2M	95,68

Para o experimento com o conjunto de dados *Fashion MNIST* foi utilizado aumento de dados com inserção de bordas nas imagens com 4 *pixels* de espessura seguido de um corte aleatório do tamanho da imagem original e um espelhamento horizontal realizado com probabilidade de 50%. Na Tabela 5.9 são apresentados os resultados obtidos pela função LDMAT juntamente com os resultados da função de custo *local loss* ([Nøkland & Eidnes, 2019](#)). Nesta tabela foi inserido o número de parâmetros das arquiteturas utilizadas porque *local loss* utiliza uma arquitetura com mais parâmetros que a LiuNet-A. A melhora que LDMAT apresenta em relação a *softmax loss* é de 0,96%, já a melhora do uso de um conjunto de soluções em relação à solução única

com a LDMAT é de 0,38%. Em relação ao melhor resultado publicado, *local loss* (Nøklund & Eidnes, 2019), o conjunto de soluções com LDMAT apresenta uma melhora de 0,33%.

Tabela 5.10: Resultados publicados com o conjunto de dados CIFAR10 e os resultados obtidos com a função de custo LDMAT. A indicação + sinaliza o uso de aumento de dados na base de dados.

Método	Arquitetura	CIFAR10	CIFAR10+
Constrative Hadsell <i>et al.</i> (2006)	LeNet	-	91,39
Triplet Schroff <i>et al.</i> (2015)	Inception	-	90,76
Multiloss Xu <i>et al.</i> (2016)	NiN	90,45	91,88
L-Softmax Liu <i>et al.</i> (2016)	LiuNet-B	92,42	94,08
SM-Softmax Liang <i>et al.</i> (2017)	LiuNet-B	92,50	94,27
SCNet Zhou <i>et al.</i> (2019)	LiuNet-B	-	92,40
PGL Qin <i>et al.</i> (2020)	LiuNet-B	92,83	94,38
GCPL Yang <i>et al.</i> (2018)	ResNet-32	92,63	-
EM-Softmax Wang <i>et al.</i> (2018b)	LiuNet-B	93,31	95,02
Anchor Loss Ryou <i>et al.</i> (2019)	ResNet-110	-	94,17
LP Loss Liang <i>et al.</i> (2020)	ResNet-110	-	94,31
C. Center Loss (Shi <i>et al.</i> , 2021b)	ResNet-18	-	94,83
Local Loss Nøklund & Eidnes (2019)	VGG-11B	-	94,70
Softmax Loss	LiuNet-B	-	92,71
LDMAT	LiuNet-B(s/FC)	-	94,60
LDMAT-Conjunto	LiuNet-B(s/FC)	-	95,64

Na Tabela 5.10 são apresentados os resultados da função de custo LDMAT em comparação com os demais resultados publicados na literatura que utilizam o conjunto de dados CIFAR10. A função custo LDMAT quando utiliza o conjunto de soluções apresenta resultados superiores a todos os demais utilizados na comparação, inclusive nos trabalhos que utilizam outras arquiteturas de CNN como a Resnet (He *et al.*, 2016). A indicação + na Tabela 5.10 representa resultados obtidos com aumento de dados padrão utilizado nas publicações: inserção de bordas de 4 *pixels* na imagem de entrada seguida de um corte aleatório do tamanho da imagem original e um espelhamento horizontal com probabilidade de 50%. O uso de aumento de dados sempre aumenta a taxa de acurácia porque traz maior variedade aos dados de entrada. A melhora que LDMAT apresenta em relação a *softmax loss* é de 1,89%. A melhora do uso de um conjunto de soluções em relação à solução única com a LDMAT é de 1,04%, já a melhora do uso de um conjunto de soluções em relação ao melhor resultado publicado, *EM-softmax* (Wang *et al.*, 2018b), é de 0,62%.

Tabela 5.11: CIFAR100

Método	Arquitetura	CIFAR100	CIFAR100+
Constrative Hadsell et al. (2006)	LeNet	-	65,96
Triplet Schroff et al. (2015)	Inception	-	63,52
Multiloss Xu et al. (2016)	NiN	65,82	68,53
L-Softmax Liu et al. (2016)	LiuNet-C	70,47	-
SM-Softmax Liang et al. (2017)	LiuNet-C	70,72	-
EM-Softmax Wang et al. (2018b)	LiuNet-C	72,21	75,69
SCNet Zhou et al. (2019)	LiuNet-C	71,72	-
PGL Qin et al. (2020)	LiuNet-C	67,32	75,69
PEDCC Zhu et al. (2020)	LiuNet-C	-	73,23
Anchor Loss Ryou et al. (2019)	ResNet110	-	74,38
LP Loss Liang et al. (2020)	ResNet110	-	74,70
C. Center Loss Shi et al. (2021b)	ResNet18	-	77,28
Local Loss Nøkland & Eidnes (2019)	VGG-11B	-	75,90
Attention Loss (Shi et al., 2021a)	ResNet18	-	78,53
Softmax Loss	LiuNet-C	-	69,02
LDMAT	LiuNet-C(s/FC)	-	74,07
LDMAT-Conjunto	LiuNet-C(s/FC)	-	78,72

O conjunto de dados CIFAR100 ([Krizhevsky, 2009](#)) possui 100 classes e somente 500 imagens de treinamento por classe, o que torna este conjunto de dados mais difícil de classificar do que o conjunto de dados CIFAR10, porém a função custo LDMAT com o uso de fusão de características apresentou resultados melhores que a maioria dos resultados publicados na literatura conforme mostra a Tabela 5.11. A indicação + na Tabela significa que o resultado foi obtido com o aumento de dados padrão, o mesmo utilizado nos outros conjuntos de dados. A melhora que LDMAT apresenta em relação a *softmax loss* é de 5,05%. A melhora do uso de um conjunto de soluções em relação à solução única com a LDMAT é de 4,65%. LDMAT com fusão das características supera, por exemplo, a função de custo *EM-softmax* que também utiliza uma estratégia de combinação de soluções, porém adota em composição com a *softmax loss*.

O conjunto de dados SVHN ([Netzer et al., 2011](#)) apresenta imagens de dígitos obtidos a partir de imagens do *Google Street View*, como se trata de números de identificação imóveis, os dígitos de menor valor têm maior ocorrência no conjunto de dados (exceto o dígito 0). Na Figura 5.4 é exibido o histograma dos rótulos das imagens da base SVHN. Os trabalhos que utilizaram este conjunto de dados optaram por utilizar o conjunto de dados extras para realizar o treinamento da CNN. Neste trabalho optou-

se por utilizar o conjunto de dados padrão. No trabalho de [Qin et al. \(2020\)](#) o conjunto de dados extra é utilizado, porém é feito um pré-processamento para que as classes fiquem balanceadas, excluiu-se imagens de classes que possuíam mais imagens que as demais classes. Na Tabela 5.12 são apresentados os resultados da função de custo LDMAT no conjunto de dados SVHN. A melhora que LDMAT apresenta em relação a *softmax loss* é de 1,25%. A melhora do uso de um conjunto de soluções em relação à solução única com a LDMAT é de 0,12%. Neste trabalho não foi utilizada a base de dados extras, portanto LDMAT foi treinada com menos imagens que os trabalhos de PGL ([Qin et al., 2020](#)), *LP loss* ([Liang et al., 2020](#)) e *local loss* ([Nøkland & Eidnes, 2019](#)).

Figura 5.4: Histograma dos rótulos do conjunto de imagens SVHN, referente a partição de treinamento e sem uso do conjunto de dados extras.

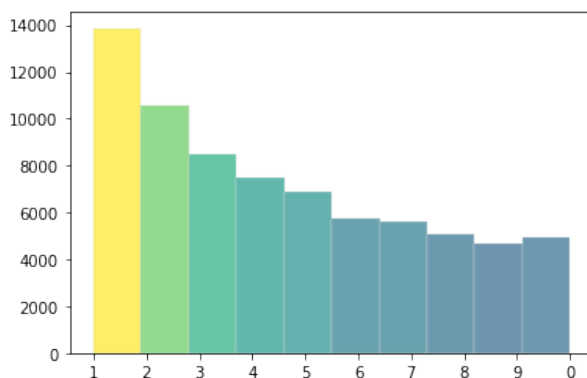


Tabela 5.12: Resultados com conjunto de dados SVHN. A coluna SVHN-ext, indica que o resultado foi obtido com a utilização do conjunto de dados extras para o treinamento.

Método	Arquitetura	SVHN	SVHN-ext
PGL Qin et al. (2020)	LiuNet-B	-	97,94
LP Loss Liang et al. (2020)	NiN	-	98,11
Local Loss Nøkland & Eidnes (2019)	VGG-8B	-	98,26
Softmax Loss	LiuNet-B	95,75	-
LDMAT	LiuNet-B(s/FC)	97,25	-
LDMAT-Conjunto	LiuNet-B(s/FC)	97,37	-

5.5 Discussão

Apesar da função de custo LDMAT ter sido testada somente com a arquitetura LiuNet, os resultados foram superiores com alguns trabalhos que utilizaram outras ar-

quitaturas como a Resnet e VGG8. A função de custo proposta tem o potencial de ser utilizada em qualquer arquitetura de rede neural que realize o treinamento em lotes.

Os diversos conjuntos de dados testaram a função de custo em diversas situações, como imagens de dígitos, nos conjuntos de dados MNIST, EMNIST e SVHN, imagens naturais como CIFAR10 e CIFAR100. LDMAT apresentou resultados robustos com a utilização de 100 classes, com o conjunto de dados CIFAR100, e para conjuntos de dados desbalanceados como SVHN. Além disso, LDMAT não impõe a necessidade de ordenação de amostras.

Muitos trabalhos na literatura utilizam a arquitetura da CNN Resnet por ter menos parâmetros para treinar e tratar o problema do desaparecimento do gradiente por meio dos blocos residuais. Apesar de possuir aproximadamente seis vezes menos parâmetros que a LiuNet (baseada na rede VGG) o tempo de treinamento da Resnet18 é de 82 segundos por época, enquanto a LiuNetB gasta aproximadamente 80s, isto ocorre que as operações de soma de camadas nos blocos residuais é custosa computacionalmente.

O trabalho de [Nøkland & Eidnes \(2019\)](#) utiliza a rede VGG8 que possui cinco vezes mais parâmetros que a arquitetura utilizada com a LDMAT com combinação de características e apresenta resultados inferiores no conjunto de dados FMNIST. Isto demonstra que a função de custo influencia muito na melhor utilização dos parâmetros da CNN para a extração das características. Um método robusto de regularização combinado com uma função de custo adequada, como a LDMAT utiliza, pode suprimir a necessidade de se utilizar muitos filtros ou muitas camadas em profundidade para se ter o efeito de redundância na extração de características.

A perturbação na matriz de rótulos tem resultados similares a técnica de desligamento de conexões (*dropout*). O uso da perturbação e melhora o resultado em uma arquitetura sem uso de *dropout*. A perturbação na matriz de rótulos, incorpora a regularização no cálculo do erro da função de custo da mesma forma como é feito com a suavização de rótulos utilizada na arquitetura da CNN *Inception-V3* ([Szegedy et al., 2016](#)). A separação da etapa de extração de característica permite de forma simples realizar a fusão de características, o que possibilita combinar diversos níveis de regularização com desligamento de conexões e perturbação na matriz de rótulos.

A desvantagem de LDMAT em relação às demais de propostas de função de custo é o número de épocas necessárias para a convergência. Enquanto algumas funções de custo apresentam os resultados com cerca de 300 épocas de treinamento, a função de custo LDMAT foi treinada com 1000 épocas. A dificuldade de convergência é relatada por trabalhos que utilizam métricas de distâncias entre amostras, talvez seja por este motivo que muitos autores optam por combinar a *softmax loss* com um termo para melhorar a compactação intraclasse das características. Acredita-se que por não ter a restrição de separação angular utilizada pela *softmax loss*, a função de custo

LDMAT possa induzir um melhor espalhamento das amostras no espaço de projeção das características. O parâmetro de distância interclasses utilizado na formulação da LDMAT (ρ) contribui para aumentar este espaço de projeção das características, enquanto que alguns trabalhos que refinam a *softmax loss* inserem uma margem de separação.

Capítulo 6

Conclusões

A busca por uma função de custo que realize a separação das classes e ao mesmo tempo tenha uma boa compacidade intraclasse das características é o objetivo de diversos trabalhos publicados na literatura. Para resolver este problema, a função de custo proposta nesta tese, utilizou matrizes de distâncias, dado que o objetivo da função consiste em aproximar uma matriz de distâncias onde amostras da mesma classes estão próximas no espaço projetado e amostras de classes diferentes estão distantes neste mesmo espaço.

Propostas de funções de custo que se baseiam na *softmax loss* são recorrentes na literatura pela sua característica de rápida convergência, porém esta função de custo induz um espaço de separação angular baseado na distância do cosseno o que dificulta a generalização do modelo. Outras métricas de distâncias, como a euclidiana, são adotadas com frequência nestes trabalhos que a combinam com a *softmax loss* para melhorar a compactação intraclasse. Entretanto, neste trabalho, utilizou-se a distância euclidiana, de forma independente da *softmax loss*, para o cálculo das distâncias entre amostras que compõem a matriz de distâncias, que apesar de permitir uma melhor compactação intraclasse das características, têm convergência mais lenta quando utilizada individualmente na função de custo.

Considera-se que a principal contribuição do trabalho são as bases teóricas para o uso de matrizes de distâncias na composição de funções de custo com a possibilidade de realizar na própria função a regularização do treinamento. A formulação da LDMAT não restringe o uso em um tipo específico de arquitetura de rede neural, cabendo o devido ajuste dos hiperparâmetros e protocolo de treinamento.

A abordagem de regularização com a perturbação na matriz de distâncias de rótulos, que é o objetivo de aproximação da função de custo LDMAT, apresentou resultados similares ao *dropout* e como esta está inserida na própria função de custo, pode ser combinada com outras estratégias de regularização como o próprio *dropout* e o decaimento de pesos. Entretanto os resultados expressivos foram alcançados com a combinação de características obtidas em diversas execuções de treinamento, com

diferentes parâmetros de regularização, isto implica um tempo maior gasto no treinamento das várias execuções e um tempo maior na fase de predição porque necessita-se realizar a fusão das características.

A separação da etapa de treinamento da etapa de classificação permitiu projetar uma função de custo que trabalha somente no espaço de projeção das características, o que tornou possível as abordagens de regularização e combinação de soluções. Entretanto tem-se o custo do projeto do classificador que não é feito de forma integrada aos filtros convolucionais.

Os resultados experimentais mostraram que a função de custo LDMAT consegue realizar uma boa generalização do modelo treinado a partir dos testes nos conjuntos de dados utilizados. Nos conjuntos de dados FMNIST, EMNIST, CIFAR10 e CIFAR100 os resultados foram superiores aos publicados, inclusive com arquiteturas diferentes das utilizadas com LDMAT. No conjunto de dados MNIST os resultados ficaram próximos aos publicados na literatura.

6.1 Trabalhos Futuros

Como continuação deste trabalho propõem-se explorar os seguintes pontos:

- Desenvolver uma função de custo a nível de bloco convolucional para aplicar penalidades específicas para parte da extração de características com os filtros convolucionais.
- Utilizar outras métricas para a aproximação das matrizes de distâncias como o *Structural Similarity Index - SSIM* (Wang *et al.*, 2004).
- Explorar a utilização de LDMAT em outras de tarefas de aprendizado de máquina.
- Explorar outras funções para realizar a perturbação na matriz de distâncias de rótulos.
- Melhorar o tempo de treinamento da função de custo LDMAT, explorando novas formas de iniciar os pesos e transferência de aprendizagem
- Aplicar a função de custo em bases maiores como a ImageNet.
- Explorar um classificador com rejeição e classes incrementais.
- Desenvolver um protocolo de treinamento para outras arquiteturas de CNNs.

Referências Bibliográficas

- BELKINA, A.C., CICCOLELLA, C.O., ANNO, R., HALPERT, R., SPIDLEN, J. & SNYDER-CAPPIONE, J.E. (2019). Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature communications*, **10**, 1–12. [vii](#), [52](#)
- CHANG, X., NIE, F., WANG, S., YANG, Y., ZHOU, X. & ZHANG, C. (2016). Compound rank-k projections for bilinear analysis. *IEEE Transactions on Neural Networks and Learning Systems*, **27**, 1502–1513. [15](#)
- CHOLLET, F. (2016). Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1800–1807. [vi](#), [32](#)
- COHEN, G., AFSHAR, S., TAPSON, J. & SCHAIK, A.V. (2017). Emnist: Extending mnist to handwritten letters. *2017 International Joint Conference on Neural Networks (IJCNN)*. [58](#)
- CYBENKO, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, **2**, 303–314. [14](#)
- DALAL, N. & TRIGGS, B. (2005). Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, **1**, 886–893. [15](#)
- FUKUSHIMA, K. (1975). Cognitron: A self-organizing multilayer neural network. *Biological Cybernetics*, **20**, 121–136. [20](#)
- FUKUSHIMA, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, **36**, 193–202. [21](#)
- GLOROT, X. & BENGIO, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Y.W. Teh & M. Titterton, eds., *Proceedings of the*

- Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9 of *Proceedings of Machine Learning Research*, 249–256, JMLR Workshop and Conference Proceedings, Sardinia, Italy. 28
- GU, J., WANG, Z., KUEN, J., MA, L., SHAHROUDY, A., SHUAI, B., LIU, T., WANG, X., WANG, L., WANG, G., CAI, J. & CHEN, T. (2017). Recent advances in convolutional neural networks. 24, 25, 27, 28, 29, 30, 31, 32, 33
- HADSELL, R., CHOPRA, S. & LECUN, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 1735–1742. 17, 36, 46, 56, 67, 69, 70
- HAN, D., KIM, J. & KIM, J. (2017). Deep pyramidal residual networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6307–6315. 32
- HE, K., ZHANG, X., REN, S. & SUN, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 63
- HE, K., ZHANG, X., REN, S. & SUN, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. vi, 14, 16, 29, 35, 69
- HINTON, G.E., SRIVASTAVA, N., KRIZHEVSKY, A., SUTSKEVER, I. & SALAKHUTDINOV, R.R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*. 23
- HU, J., SHEN, L. & SUN, G. (2018). Squeeze-and-excitation networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7132–7141. vi, 33
- HU, Y., WEN, G., LUO, M., DAI, D., MA, J. & YU, Z. (2018). Competitive inner-imaging squeeze and excitation for residual network. *arXiv preprint arXiv:1807.08920*. 33
- HUANG, G., LIU, Z., v. D. MAATEN, L. & WEINBERGER, K.Q. (2017). Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261–2269. vi, 30
- HUBEL, D.H. & WIESEL, T.N. (1959). Receptive fields of single neurons in the cat's striate cortex. *Journal of Physiology*, 148, 574–591. 20
- IOFFE, S. & SZEGEDY, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning - ICML*, vol. 37, 448–456. 16, 27, 60

- IVAKHNENKO, A.G. (1971). Polynomial theory of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 364–378. 20
- JABBAR, A., LI, X. & OMAR, B. (2021). A survey on generative adversarial networks: Variants, applications, and training. *ACM Computing Surveys*, 54. 14
- JIN, N., WU, J., MA, X., YAN, K. & MO, Y. (2020). Multi-task learning model based on multi-scale cnn and lstm for sentiment classification. *IEEE Access*, 8, 77060–77072. 14
- KRIZHEVSKY, A. (2009). Learning multiple layers of features from tiny images. *Technical Report [Online]*. Available: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>. 58, 70
- KRIZHEVSKY, A., SUTSKEVER, I. & HINTON, G.E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou & K.Q. Weinberger, eds., *Advances in Neural Information Processing Systems 25*, 1097–1105, Curran Associates, Inc. 14, 23, 35
- KULLBACK, S. & LEIBLER, R.A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22, 79–86. 16, 35
- LECUN, Y., BOSER, B., DENKER, J.S., HENDERSON, D., HOWARD, R.E., HUBBARD, W. & JACKEL, L.D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1, 541–551. 21, 57
- LECUN, Y., BOTTOU, L., BENGIO, Y. & HAFNER, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 2278–2324. vi, 15, 21
- LECUN, Y., CORTES, C. & BURGES, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, 2. 58
- LI, Z., LIU, F., YANG, W., PENG, S. & ZHOU, J. (2021). A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21. 35
- LIANG, C., ZHANG, H., YUAN, D. & ZHANG, M. (2020). Location property of convolutional neural networks for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15. vi, 39, 44, 47, 69, 70, 71
- LIANG, X., WANG, X., LEI, Z., LIAO, S. & LI, S.Z. (2017). Soft-margin softmax for deep classification. In D. Liu, S. Xie, Y. Li, D. Zhao & E.S.M. El-Alfy, eds., *Neural Information Processing*, 413–421, Springer International Publishing, Cham. 38, 46, 60, 67, 69, 70

- LIN, M., CHEN, Q. & YAN, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*. 25
- LIU, W., WEN, Y., YU, Z. & YANG, M. (2016). Large-margin softmax loss for convolutional neural networks. In M.F. Balcan & K.Q. Weinberger, eds., *International Conference on Machine Learning - ICML*, vol. 48, 507–516, JMLR.org. vi, ix, 16, 36, 38, 39, 46, 60, 61, 63, 67, 69, 70
- LOSHCHILOV, I. & HUTTER, F. (2017). Sgdr: stochastic gradient descent with warm restarts. In *International conference on learning representations ICLR*. 63
- LOWE, D.G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 91–110. 15
- LUO, Y., WONG, Y., KANKANHALLI, M. & ZHAO, Q. (2020). \mathcal{G} -softmax: Improving intraclass compactness and interclass separability of features. *IEEE Transactions on Neural Networks and Learning Systems*, 31, 685–699. 39, 46
- MCCULLOCH, W.S. & PITTS, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115–133. 20
- NETZER, Y., WANG, T., COATES, A., BISSACCO, A., WU, B. & NG, A.Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. 58, 70
- NØKLAND, A. & EIDNES, L.H. (2019). Training neural networks with local error signals. In K. Chaudhuri & R. Salakhutdinov, eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, vol. 97 of *Proceedings of Machine Learning Research*, 4839–4850, PMLR. vii, 43, 44, 47, 67, 68, 69, 70, 71, 72
- P. GOYAL, R.G.P.N.L.W.A.K.A.T.Y.J., P. DOLLAR & HE, K. (2017). Accurate, large mini-batch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*. 63
- PENG, H. & YU, S. (2021). Beyond softmax loss: Intra-concentration and inter-separability loss for classification. *Neurocomputing*, 438, 155–164. 16, 36
- QIN, Y., YAN, C., LIU, G., LI, Z. & JIANG, C. (2020). Pairwise gaussian loss for convolutional neural networks. *IEEE Transactions on Industrial Informatics*, 16, 6324–6333. vi, 17, 36, 42, 43, 47, 60, 67, 68, 69, 70, 71
- QUEIROZ, F., BRAGA, A. & PEDRYCZ, W. (2009). Sorted kernel matrices as cluster validity indexes. In *IFSA/European Society for Fuzzy Logic and Technology Conference*, 1490–1495. 18

- RANJAN, R., CASTILLO, C.D. & CHELLAPPA, R. (2017). L2-constrained softmax loss for discriminative face verification. *arXiv preprint arXiv:1703.09507*. vi, 17, 38, 39, 46
- RIESENHUBER, M. & POGGIO, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2, 1019–1025. 20
- ROSASCO, L., DE VITO, E., CAPONNETTO, A., PIANA, M. & VERRI, A. (2004). Are loss functions all the same? *Neural Computation*, 16, 1063–1076. 35
- ROSENBLATT, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65–386. 20
- RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M., BERG, A. & LI, F.F. (2014). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115. 15
- RYOU, S., JEONG, S.G. & PERONA, P. (2019). Anchor loss: Modulating loss scale based on prediction difficulty. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 45, 47, 69, 70
- SCHROFF, F., KALENICHENKO, D. & PHILBIN, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 815–823, IEEE Computer Society. 17, 37, 46, 67, 69, 70
- SHI, X., XING, F., XU, K., CHEN, P., LIANG, Y., LU, Z. & GUO, Z. (2021a). Loss-based attention for interpreting image-level prediction of convolutional neural networks. *IEEE Transactions on Image Processing*, 30, 1662–1675. 70
- SHI, Z., WANG, H. & LEUNG, C.S. (2021b). Constrained center loss for convolutional neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 1–9. 41, 46, 69, 70
- SHORTEN, C. & KHOSHGOFTAAR, T.M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6, 1–48. 16
- SIAGIAN, C. & ITTI, L. (2007). Rapid biologically-inspired scene classification using features shared with visual attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29, 300–312. 20
- SILVESTRE, L.J., LEMOS, A.P., BRAGA, J.P. & BRAGA, A.P. (2015). Dataset structure as prior information for parameter-free regularization of extreme learning machines. *Neurocomputing*, 169, 288–294. 18

- SIMONYAN, K. & ZISSERMAN, A. (2015). Very deep convolutional networks for large-scale image recognition. In Y. Bengio & Y. LeCun, eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 14, 24, 60
- SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I. & SALAKHUTDINOV, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 1929–1958. vi, 16, 24
- SRIVASTAVA, R.K., GREFF, K. & SCHMIDHUBER, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*. 28
- SUN, Y.K., LIANG, D., WANG, X. & TANG, X. (2015). Deepid3: Face recognition with very deep neural networks. *ArXiv*, abs/1502.00873. 37
- SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V. & RABINOVICH, A. (2015). Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*. vi, 26
- SZEGEDY, C., VANHOUCKE, V., IOFFE, S., SHLENS, J. & WOJNA, Z. (2016). Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818–2826, IEEE Computer Society, Los Alamitos, CA, USA. vi, 16, 27, 49, 72
- SZEGEDY, C., IOFFE, S., VANHOUCKE, V. & ALEMI, A.A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI'17*, 4278–4284, AAAI Press. 29
- TIELEMAN, T. & HINTON, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning. 27
- WANG, F., CHENG, J., LIU, W. & LIU, H. (2018a). Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25, 926–930. vi, 16, 39, 41, 46
- WANG, Q., QIN, Z., NIE, F. & LI, X. (2021). C2dnda: A deep framework for nonlinear dimensionality reduction. *IEEE Transactions on Industrial Electronics*, 68, 1684–1694. 15
- WANG, X., ZHANG, S., LEI, Z., LIU, S., GUO, X. & LI, S.Z. (2018b). Ensemble soft-margin softmax loss for image classification. In J. Lang, ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 992–998, ijcai.org. 42, 47, 55, 60, 67, 69, 70

- WANG, Z., BOVIK, A., SHEIKH, H. & SIMONCELLI, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, **13**, 600–612. [75](#)
- WEN, Y., ZHANG, K., LI, Z. & QIAO, Y. (2016). A discriminative feature learning approach for deep face recognition. In B. Leibe, J. Matas, N. Sebe & M. Welling, eds., *ECCV (7)*, vol. 9911 of *Lecture Notes in Computer Science*, 499–515, Springer. [40](#), [46](#)
- XIAO, H., RASUL, K. & VOLLGRAF, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*. [57](#), [58](#)
- XIE, S., GIRSHICK, R., DOLLAR, P., TU, Z. & HE, K. (2017). Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5987–5995. [vi](#), [29](#), [30](#)
- XU, C., LU, C., LIANG, X., GAO, J., ZHENG, W., WANG, T. & YAN, S. (2016). Multi-loss regularized deep neural network. *IEEE Transactions on Circuits and Systems for Video Technology*, **26**, 2273–2283. [17](#), [18](#), [44](#), [47](#), [55](#), [67](#), [69](#), [70](#)
- YANG, H., ZHANG, X., YIN, F. & LIU, C. (2018). Robust classification with convolutional prototype learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3474–34826, Computer Vision Foundation / IEEE Computer Society. [18](#), [40](#), [47](#), [56](#), [67](#), [69](#)
- ZAGORUYKO, S. & KOMODAKIS, N. (2016). Wide residual networks. In *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. [31](#)
- ZEILER, M.D. & FERGUS, R. (2014). Visualizing and understanding convolutional networks. In D. Fleet, T. Pajdla, B. Schiele & T. Tuytelaars, eds., *Computer Vision – ECCV 2014*, 818–833, Springer International Publishing, Cham. [vi](#), [24](#), [25](#)
- ZHANG, Z. (2016). Derivation of backpropagation in convolutional neural network (cnn). *University of Tennessee, Knoxville, TN*. [22](#), [23](#)
- ZHONG, Z., ZHENG, L., KANG, G., LI, S. & YANG, Y. (2020). Random erasing data augmentation. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI'20*, 13001–13008, AAAI Press. [16](#)
- ZHOU, L., WANG, Z., LUO, Y. & XIONG, Z. (2019). Separability and compactness network for image recognition and superresolution. *IEEE Transactions on Neural Networks and Learning Systems*, **30**, 3275–3286. [17](#), [42](#), [46](#), [60](#), [67](#), [69](#), [70](#)

ZHU, Q., ZHANG, P., WANG, Z. & YE, X. (2020). A new loss function for cnn classifier based on predefined evenly-distributed class centroids. *IEEE Access*, **8**, 10888–10895. [41](#), [46](#), [60](#), [68](#), [70](#)